MODELING NETWORK INTERDICTION TASKS

DISSERTATION

Benjamin S. Kallemyn, Major, USAF

AFIT-ENS-DS-15-S-032

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government. This is an academic work and should not be used to imply or infer actual mission capability or limitations.

AFIT-ENS-DS-15-S-032

MODELING NETWORK INTERDICTION TASKS

DISSERTATION

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Operations Research

Benjamin S. Kallemyn, BS, MS

Major, USAF

September 2015

AFIT-ENS-DS-15-S-032

MODELING NETWORK INTERDICTION TASKS

Benjamin S. Kallemyn, BS, MS
Major, USAF

Committee Membership:

Richard F. Deckro, DBA
Chairman

LTC Brian J. Lunday, PhD
Member

Lt Col Matthew J. Robbins, PhD
Member

Maj Dustin G. Mixon, PhD
Member

ADEDEJI B. BADIRU, PhD
Dean, Graduate School of
Engineering and Management

AFIT-ENS-DS-15-S-032

Abstract

Mission planners seek to target nodes and/or arcs in networks that have the greatest benefit for an operational plan. In joint interdiction doctrine, a top priority is to assess and target the enemy's vulnerabilities resulting in a significant effect on its forces.

An interdiction task is an event that targets the nodes and/or arcs of a network resulting in its capabilities being destroyed, diverted, disrupted, or delayed. Lessons learned from studying network interdiction model outcomes help to inform attack and/or defense strategies.

A suite of network interdiction models and measures is developed to assist decision makers in identifying critical nodes and/or arcs to support deliberate and rapid planning and analysis. The interdiction benefit of a node or arc is a measure of the impact an interdiction task against it has on the residual network.

The research objective is achieved with a two-fold approach. The measures approach begins with a network and uses node and/or arc measures to assess the benefit of each for interdiction. Concurrently, the models approach employs optimization models to explicitly determine the nodes and/or arcs that are most important to the planned interdiction task.

I would like to thank the wife, and the kids.

Most of all, I give thanks to The Way, The Truth, and The Life.

Acknowledgements

I would like to thank the advisor, the committee, and the wingman.

My advisor, Dr. Deckro, was instrumental in helping to frame the dissertation into a cohesive document and gave immeasurable advice through the entire process.

My committee, Dr. Lunday, Dr. Robbins, and Dr. Mixon, was always available to improve my work; their time and guidance were invaluable.

Finally, my wingman, Smalz, during this second round at AFIT, helped make the time more bearable, even if it was sometimes less productive.

Benjamin S. Kallemyn

Table of Contents

Page

Page

List of Figures

Figure Page

List of Tables

Table Page

Table Page

Table Page

MODELING NETWORK
INTERDICTION TASKS

I. Introduction

1.1 Background

Today's world is highly connected, and technology advances allow it to be more connected every day. Transportation networks comprised of roads, railroads, air travel routes, and waterways allow people and resources to traverse the world. Telecommunications networks consisting of telephone lines, cellular towers, and satellite relays allow professionals and families to stay in touch through voice or video calls. Power grids provide electricity to homes, offices, and recreational facilities. Social networks, whether through online tools such as Facebook and Twitter or via real connections such as family and work relationships, connect people around the world. Other networks exist that connect people or things to resources they require.

The technological advancement of these networks has brought improvements to quality of life for people around the world [39]. These networks have allowed better medical care which extends life expectancies [39]. More efficient methods for delivering clean water have saved many lives as well [50]. Unfortunately, these technological advances have left many vulnerabilities that adversaries may attempt to exploit.

When a weakness in one of these networks is detected by a malicious actor, they may attack the flaw to cause widespread chaos. Attacks such as these are considered a form of network interdiction. Assessing the networks over which a governing body has control is imperative for identifying these vulnerabilities and hardening them against such attacks.

Alternatively, when considering offensive actions against an enemy's infrastructure networks, similar analysis may identify their weaknesses to attack so allied forces can gain the advantage.

Network interdiction models are used to inform attack strategies against a network (*i.e.*, transportation, telecommunications, power grid, social, and so forth) to reduce its ability to function at peak performance. Conversely, the lessons learned from analyzing such attacks shed light on how to best defend the same network. In either case, models developed for this purpose will prove useful.

The US Department of Defense defines interdiction as "actions to divert, disrupt, delay, or destroy" capabilities "to prevent the adversary from using assets at the time and place of his choosing." [1:p. I-1–I-2]. In a general context, network interdiction involves performing one or more of these interdiction activities to a network and measuring the effect. Historical examples of network interdiction applied in warfare are documented as early as 479 BC; Herodotus records the Battle of Plataea where the Persian army attacked Greek supply lines and disrupted Greek water access [41]. More recently, Israeli attacks in Gaza targeted power plants, Hamas leadership, and the television network controlled by Hamas [2]. These and other examples reveal the act of attacking the enemy's supply lines and support infrastructure can have a substantial impact on the outcome of battles [69:p. 1].

In this dissertation, the term *interdiction task* is used to describe an event that targets the nodes and/or arcs of a network resulting in its capabilities being destroyed, disrupted, delayed, or diverted. Network interdiction models can be used to analyze the impact of such a task against a network. Lessons learned from studying network interdiction model outcomes help to inform attack strategies by identifying nodes or arcs in the network that, when attacked, provide a tactical advantage by destroying, disrupting, delaying, or diverting the adversary's capabilities. Conversely, the same models can be used to help identify critical nodes or arcs in networks that must be defended and/or hardened.

The Department of Defense publishes doctrine to govern the practice of warfare. In the doctrine pertaining to the fundamentals of joint interdiction, optimal targeting is of utmost importance.

> A key task during interdiction planning is analyzing the enemy for critical vulnerabilities that, if attacked, will have a disruptive effect across significant portions of the enemy force. [1:p. I-3]

This research effort will assist decision makers in identifying critical nodes and/or arcs. Such knowledge will aid in the planning and execution of interdiction operations against an adversary's network.

1.2 Research Overview

Problem Statement. Given a network, identify the arcs and/or nodes to target that are most effective in destroying, disrupting, diverting, or delaying its capabilities.

Research Objective. Develop a suite of network models and measures that will assist decision makers in identifying critical nodes and/or arcs to support deliberate and rapid planning and analysis.

The interdiction benefit of a node or arc is a measure of the impact an interdiction task against it has on the residual network. The nodes and/or arcs with the largest benefit are those that should be targeted when developing an attack strategy against a network or defended (and/or hardened) when developing a defense strategy. Interdiction benefit is mission specific. An interdiction task that targets a specific node via a surgical strike may be more beneficial compared to an alternative task due to the resulting narrow impact and lesser collateral damage or cascading effects. Alternatively, an interdiction task may be preferable to an alternative if it leads to widespread effects cascading throughout a network. The suite

of models and measures developed in this dissertation can be attuned to model either of these mission-specific perspectives of interdiction benefit.

The interdiction benefit is assessed from one of two perspectives. First, it is determined using measures based on the topology of the infrastructure network. Alternatively, it is determined explicitly using the solutions to optimization models. These two approaches are illustrated in Figure 1 and are referenced in the remainder of this research using the following terminology and description.

Measures Approach. Begin with a network and use node and/or arc measures to assess the benefit of each for interdiction. The fields of graph theory and social network analysis utilize measures to indicate various features of nodes and arcs. This research will both identify the currently used network measures and introduce new measures that indicate the interdiction benefit of a node or arc.

This approach provides a suite of tools that allows an analyst to nominate candidate nodes to target for the largest impact on the network in question or for the least impact. It will demonstrate that the time to compute network measures can be completed rapidly when geodesic (*i.e.*, shortest path length) information is retained, that extending the information stored allows the rapid calculation of the network measures, and that utilizing this information allows the assessment of each node's removal from the network.

Models Approach. Employ optimization models to explicitly determine the nodes and/or arcs that are most important to the planned interdiction task. This research will review or propose models related to the four interdiction tasks (*i.e.*, destroy, disrupt, divert, and delay) on networks. The solutions to these models will identify the nodes and/or arcs with the largest interdiction benefit.

This approach provides the analyst and decision maker an array of modeling tools and options to utilize when investigating options for planning and executing interdiction oper-

ations. The suite of models apply each of the four interdiction tasks separately and are developed to extend to a variety of additional mission constraints and options.

The depiction of the framework in Figure 1 will be used in the remainder of this document to denote where the discussion fits within this research framework. The four interdiction tasks identified in joint doctrine, and depicted on the framework chart, are defined as follows.

Destroy

> actions that "damage the structure, function, or condition of a target so that it can neither perform as intended nor be restored to a usable condition, rendering it ineffective or useless" [1:p. I-4]

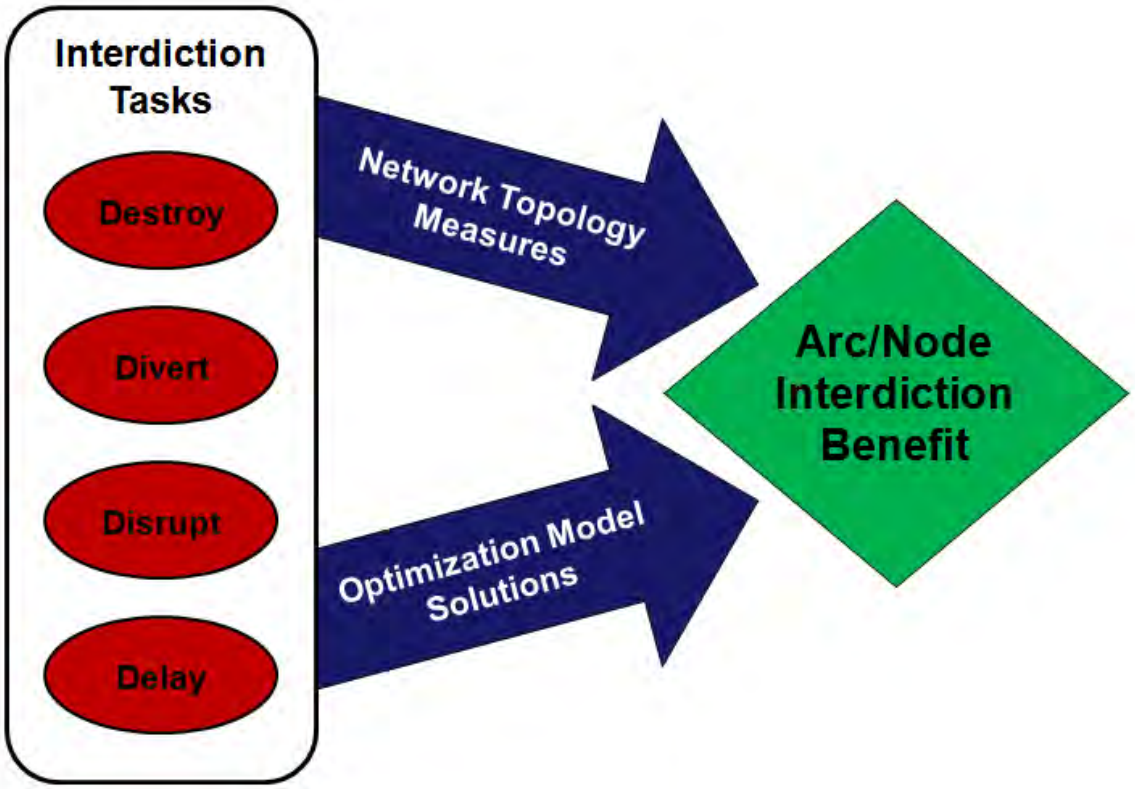


Figure 1. Research Framework

Divert

actions that "divert enemy forces or assets from areas where there are critical operational requirements for them" [1:p. I-2]

Disrupt

actions that "interrupt or impede enemy or enemy capabilities or systems" [1:p. I-2]

Delay

actions that "delay the time of arrival of enemy forces or capabilities" [1:p. I-3]

Each of the interdiction tasks is used to fulfill the purpose given in the definition.

The research objectives and contributions align with the two approaches. They are listed with the objectives and contributions related to the measures approach first, and then those related to the models approach.

- Generate a network algorithm that readily computes and/or updates measures when a node is destroyed.
- Develop a network interdiction modeling framework for considering:
  - Destroying nodes.
  - Diverting network resources from traversing through any of a predefined set of nodes.
  - Disrupting capabilities based on partial damage.
  - Delaying the restoration of network resources.

The result of this research thrust is a suite of network interdiction models and measures that will assist decision makers in identifying critical nodes and/or arcs. This array of measures and models may serve as modeling options for offensive and defensive operations. The operations that can be considered when utilizing this suite of measures and models may

be either kinetic or non-kinetic actions such as detailed observation, signals collection, denial of service, or possibly destruction. Thus, the set of models and measures developed in this dissertation provide a foundation for analysis of operational offensive and/or defensive plans and a basis for future research.

1.3 Dissertation Overview

The remainder of this dissertation is organized as follows. Chapter II reviews pertinent literature relevant to the problem statement and research objectives. Chapter III demonstrates the utility of measures for selecting nodes for destruction. Chapter IV proposes new models that represent the destruction of nodes and/or arcs in a network. Chapter V introduces the network diverting problem and models to solve it. Chapter VI demonstrates the utility of a number of models for network disrupting. Chapter VII presents a modeling framework to represent interdiction delaying tasks. Finally, Chapter VIII provides a summary of the research contributions and avenues of future research.

II. Pertinent Literature

2.1 Introduction

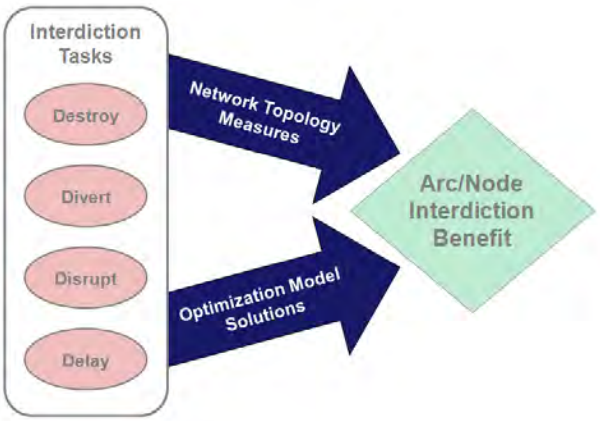


Figure 2. Research Framework: Pertinent Literature

This chapter summarizes pertinent literature and forms a foundation for the techniques of the measures and models approaches of the research framework, which is depicted in Figure 2. The measures approach has its foundation in the fields of social network analysis (SNA) and graph theory. Each of these fields uses measures to assess an attribute of the network topology. Section 2.2 reviews literature pertinent to this approach. The models approach utilizes optimization model solutions to determine the most important nodes and/or arcs. Optimization models that are foundational for understanding the four interdiction tasks are reviewed in Section 2.3.

2.2 Network Topology Measures

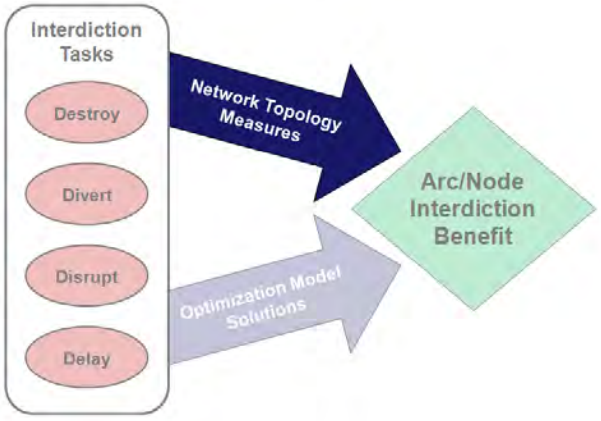


Figure 3. Research Framework: Network Topology Measure Literature

This section provides a review of literature focussed on determining common measures used in graph theory and social network analysis (SNA). When selecting a measure to indicate an interdiction benefit for a specified task, the feature of the network's topology that the measure indicates will be important. For each measure commonly applied in these fields, insights including a description of the measure, the way it is computed, and its common uses are provided. These insights will allow the extension of these measures to interdiction benefit.

In graph theory, there are measures that serve to illustrate features of of individual nodes and the entire network. Some are computed with ease, others are used in assumptions to make strong conclusions for graphs with the specified feature. In SNA, several measures have been used to indicate the importance of nodes in a network. A number of these measures are the same as the graph theory measures, while others are different. A subset of graph theoretic and SNA measures are presented.

2.2.1 Nodal Measures - Graph Theory.

The first measures reviewed are specific to individual nodes in a network and are common to graph theory literature. The main source is the introductory graph theory book by West [65]. The networks in Figure 4 are used to illustrate the measures and will be referenced throughout this section. Network N_2 is used to illustrate directed graphs in the work of Chartrand and Tian [13:p. 18]. Networks N_1 and N_3 are variations of their directed network.

Each of the following measures reflect properties of a network or graph, N. The network consists of a set V of vertices or nodes in which each of the n nodes is indexed $i = 1, \ldots, n$. Nodes are connected by edges or arcs. The set of arcs is denoted E and individual arcs are identified by pairs (i, j), where $i, j \in V$. An adjacency matrix A is the matrix representation of the network. A is an $n \times n$ matrix in which each entry $a_{ij} = 1$ if there is an arc from node i to node j and otherwise is 0. An undirected graph has a symmetric adjacency matrix

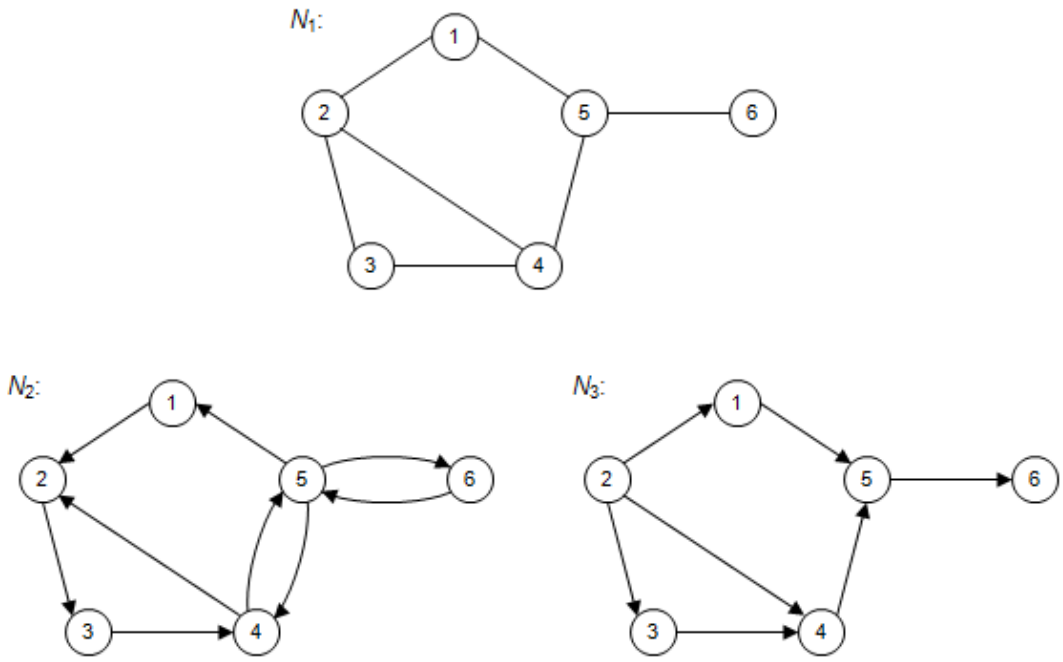


Figure 4. Undirected and directed networks [13:p. 18]

since arcs can be thought of as being connected equally in both directions or that the order of the pair does not matter [65:p. 6]. Network N_1 in Figure 4 is undirected; flow or travel may occur in ether direction along each arc. For directed networks, the order of the arc pair matters. The first node of the pair is denoted as the tail and is the starting point of the arc, the second node of the arc pair is the head and is the end point of the arc. Flow or travel along an arc in a directed network is allowed only in the direction of the arrows in the network's depiction, that is, over arcs from the tail nodes to the head nodes [65:p. 53]. An undirected network can be represented as a directed network with a separate arc for each direction between connected nodes. In Figure 4, networks N_2 and N_3 depict directed networks.

A directed network is *strongly connected* if, for every (i, j)-pair of nodes in the network, there is a directed path from node i to node j [65:p. 56]. Network N_2 in Figure 4 is strongly connected, while N_3 is *weakly connected* since there is not a directed path between every pair of nodes [65:p. 56]. For instance, there is no directed path from node 1 to node 2.

Consider a *flow network* in which arcs represent pipes where the flow is allowed to travel in one direction in a pipe. Pipe junctions are modeled by nodes, and each pipe is limited by its capacity. There are specific nodes for the source and terminus denoted s and t, respectively. If there are multiple sources (termini), a *supersource* (*superterminus*) is added with arcs to each of the sources (from the termini), creating a network with a single source and terminus [29]. Network N_3 in Figure 4 is a flow network with the source at node 2 and the terminus at node 6.

2.2.1.1 Degree.

For undirected networks, the degree $d(i)$ is the number of arcs that are incident to node i. It is computed as the sum of the row (or column) of the adjacency matrix associated with

node i,

$$d(i) = \sum_{j \in V} a_{ij} = \sum_{j \in V} a_{ji}. \tag{2.1}$$

The number of arcs in an undirected network, m, is half of the sum of the degrees of each node in the network, $m = \dfrac{\sum_{i \in V} d(i)}{2}$ [65:p. 35]. For directed networks, the degree of a node is distinguished based upon whether the node is at the head or tail of an arc. The out-degree $d^+(i)$ of a node is the number of arcs that originate at that node, whereas the in-degree $d^-(i)$ of a node is the number of arcs that terminate at that node. The out-degree is computed by summing the row of the adjacency matrix for node i, and the in-degree is the sum of the ith column of the adjacency matrix.

$$d^+(i) = \sum_{j \in V} a_{ij}, \tag{2.2}$$

$$d^-(i) = \sum_{j \in V} a_{ji}, \tag{2.3}$$

$$d(i) = d^+(i) + d^-(i). \tag{2.4}$$

Since each arc has a head and a tail, $m = \sum_{i \in V} d^+(i) = \sum_{i \in V} d^-(i)$ [65:p. 59].

Network N_1 in Figure 4 is an undirected network, so the degree calculation from (2.1) is used. For the directed networks N_2 and N_3, the degree calculations from (2.2)-(2.4) are used. Table 1 lists the degree measures for these networks. The degree of node 4 in the

	N_1	N_2			N_3		
Node (i)	$d(i)$	$d^+(i)$	$d^-(i)$	$d(i)$	$d^+(i)$	$d^-(i)$	$d(i)$
1	2	1	1	2	1	1	2
2	3	1	2	3	3	0	3
3	2	1	1	2	1	1	2
4	3	2	2	4	1	2	3
5	3	3	2	5	1	2	3
6	1	1	1	2	0	1	1

Table 1. Degree nodal measures for networks in Figure 4

directed network N_1 is 3. The in-degree of node 4 in N_2 is 2 (two arcs are directed into the node), and its out-degree is 2 (two arcs are directed out of the node), resulting in a degree of 4.

2.2.1.2 Eccentricity.

The distance between nodes in a network is measured using paths. The *eccentricity* of node i is the maximum of the shortest paths from node i to all other nodes. In other words, eccentricity of i is the length of the shortest path from node i to the node that is the farthest distance away. For undirected, connected networks, each node will have a finite eccentricity [65:p. 71]. The length of the shortest path between nodes in which there is no path is considered to be infinite. Therefore, in a directed network, it is possible that the eccentricity for some nodes may be infinite since, although the network is connected, the direction of flow does not allow a directed path between the node and another node. Sometimes an analyst is concerned with the number of arcs in the shortest path between nodes; in this case, the distance used should be of unit length, or equivalently, use the adjacency matrix as the distance matrix: $d_{ij} = a_{ij}$.

Let $d_N(i, j)$ denote the shortest (i, j)-path in an undirected network N. Then the eccentricity of a node is [65:p. 71]

$$e(i) = \max_{j \in V} d_N(i, j) = \max_{j \in V} d_N(j, i). \tag{2.5}$$

For directed networks, the eccentricity of a node is distinguished based upon whether the node is at an endpoint of the shortest path. The *out-eccentricity* $e^+(i)$ of a node is the maximum shortest path distance from node i to any other node in the network whereas the *in-eccentricity* $e^-(i)$ of a node is the maximum shortest path distance from any other node to node i [37:p.884]. The out-eccentricity, the in-eccentricity, and the eccentricity of a node

in a directed network are [46:p. 381]

$$e^+(i) = \max_{j \in V} d_N(i,j), \tag{2.6}$$

$$e^-(i) = \max_{j \in V} d_N(j,i), \tag{2.7}$$

$$e(i) = \max(e^+(i), e^-(i)). \tag{2.8}$$

Using an algorithm to find all the shortest paths makes the calculation of the measure eccentricity tractable. The Floyd-Warshall Algorithm (denoted Algorithm 1) identifies the shortest path distance between all node combinations. The implementation presented is based on the presentation in [3:p. 148]. The main contribution of the algorithm is the induction step based on dynamic programming that ensures the distance of the shortest (i,j) path is computed. This insight is attributed to Warshall [62]. The algorithm's present form is attributed to Floyd [28]. The Floyd-Warshall Algorithm, as described in Algorithm 1, runs in $O(n^3)$ time since there is an iteration through each node in the network for every node pair [3:p. 148].

The output of Algorithm 1 gives a matrix M of all shortest paths where the (i,j)th entry, m_{ij}, is the shortest path from node i to node j. For this research, $m_{ii} = 0$ by assumption. In addition, the algorithm outputs a matrix P of the predecessor nodes where the (i,j)th entry, p_{ij}, is the node predecessor on the shortest path from node i to node j. Using this information, any shortest (i,j) path can be obtained by backtracking from node j. For instance, if node k is the predecessor for the (i,j) path, *i.e.* $p_{ij} = k$, the next predecessor is determined by p_{ik} until node i is reached [3:p. 148].

The out- and in-eccentricity measures (2.6)–(2.7), and therefore the eccentricity (2.5) of a node, can be computed based on the output of Algorithm 1. The out-eccentricity for node i is the maximum value in row i of M, $e^+(i) = \max_{j \in V} m(i,j)$. The in-eccentricity for node i

Algorithm 1 Floyd-Warshall Algorithm [3:p. 148]

Input

A network with nodes $N = 1, \ldots, n$ and arcs $E = (i, j), i, j \in N$. The node adjacency matrix A and non-negative arc distances d_{ij}.

Output

A matrix M of all shortest paths where the (i, j)th entry is the shortest path from node i to node j. A matrix P of the predecessor nodes where the (i, j)th entry is the node predecessor on the shortest path from node i to node j.

Initialization:

Generate a $n \times n$ matrix M with elements m_{ij} having an infinite value: $m_{ij} = \infty$, $\forall i, j \in N$.
Ensure there are no paths from a node to itself: $m_{ii} = 0$, $\forall i \in N$.
Populate M with the known distance between adjacent nodes: $m_{ij} = d_{ij}$, $\forall (i, j) \in E$.
Populate P with the known predecessor of adjacent nodes: $p_{ij} = i$, $\forall (i, j) \in E$.

Shortest Paths:

For each k in the set of nodes,
  For each (i, j) in $N \times N$,
    If $m_{ij} > m_{ik} + m_{kj}$, then
      Update the shortest (i, j) path length, $m_{ij} = m_{ik} + m_{kj}$.
      Update the predecessor node on the shortest path, $p_{ij} = p_{kj}$.
  Next (i, j).
Next k.

---

is the maximum value in column i of M, $e^-(i) = \max_{j \in V} m(j, i)$. When referring to directed networks, the analyst must be cautious with the terminology used for eccentricity. In some cases, the out-eccentricity is the only measure used for directed networks [13, 12]. In these cases, eccentricity refers to the minimum distance from the node to all other nodes.

A node j is called an *out-centric* node of i if the shortest (i, j)-path length is equal to the out-eccentricity of node j, $d_N(i, j) = e^+(i)$. Likewise, node j is an *in-centric* node of i if $d_N(j, i) = e^-(i)$ [34].

Recall that flow networks are directed networks with a source node having only outgoing arcs and a terminus node having only incoming arcs. Thus, in a flow network, the source, or supersource if there are multiple sources, is the only node with a finite out-eccentricity. The source (supersource) is the only node from which all other nodes are reachable in a connected

	N_1			N_2			N_3
Node (i)	$e(i)$	$e^+(i)$	$e^-(i)$	$e(i)$	$e^+(i)$	$e^-(i)$	$e(i)$
1	2	5	4	5	∞	∞	∞
2	3	4	3	4	3	∞	∞
3	3	3	4	4	∞	∞	∞
4	2	2	3	3	∞	∞	∞
5	2	3	4	4	∞	∞	∞
6	3	4	5	5	∞	3	∞

Table 2. Eccentricity nodal measures for networks in Figure 4

network, *i.e.* there is a directed path from the source to all other nodes. All other nodes have infinite out-eccentricities. Conversely, the terminus node, or super terminus if there are multiple termini, is the only node with a finite in-eccentricity. The terminus (superterminus) is the only node reachable from all other nodes, *i.e.* there is a directed path from all other nodes to the terminus. The in-eccentricity for all other nodes is infinite.

Table 2 lists the eccentricity measures for the networks in Figure 4. The adjacency matrices for each network were used in Algorithm 1 (Floyd-Warshall) with unit distances between adjacent nodes (counting the number of arcs on the shortest path). In network N_1, node 2 has an eccentricity of 3 since node 6 is the maximum shortest distance from node 2 to any other node, which traverses three arcs.

2.2.1.3 Total Distance.

The *total distance* $\mathrm{td}(i)$ of node i in a network is the sum of the length of the shortest paths from node i to all other nodes in the network [12]. The total distance of a node represents how close a node is to all other nodes since it measures the sum of the distances to every other node. In a social network, the total distance may indicate how quickly information can reach all others, where it is assumed people with smaller total distance measures reach everyone more quickly. The total distance is sometimes referred to as the *status* of the node

and is computed by [13]

$$\mathrm{td}(i) = \sum_{j \in V} d_N(i, j). \tag{2.9}$$

For an undirected network, the total distance of a node can be computed from the output of Algorithm 1 as $\mathrm{td}(i) = \sum_{j \in V} m_{ij} = \sum_{i \in V} m_{ij}$. The second equality in the computation is valid because the adjacency matrix and the distance matrix, which is composed of the (i, j)-distances between adjacent nodes, are symmetric.

For directed networks, the total distance from a node to all others or from all others to the node may be different. This difference allows an analyst to distinguish the extent to which a node influences/reaches into the network, or is influenced by/connected to the network. These quantities are referred to as the *out-transmission* $\sigma^+(i)$ and *in-transmission* $\sigma^-(i)$ of node i, respectively [55:pp. 1–2]:

$$\sigma^+(i) = \sum_{j \in V} d_N(i, j), \tag{2.10}$$

$$\sigma^-(i) = \sum_{j \in V} d_N(j, i). \tag{2.11}$$

Notice that the calculation of total distance (2.9) and out-transmission (2.10) are the same. In cases where the network has no (i, j)-path, the distance is infinite, $d_N((i, j) = \infty$. In these cases, the total distance, out-transmission, or in-transmission will also be infinite. The measure will be infinite for all nodes except the source (out-transmission) or terminus (in-transmission) in flow networks with a single source and/or terminus. In networks that are not strongly connected, some nodes will have infinite out- or in-transmission measures.

The out- and in-transmission measures can be calculated based on the output matrix M from Algorithm 1 by $\sigma^+(i) = \sum_{j \in V} m_{ij}$ and $\sigma^-(i) = \sum_{j \in V} m_{ji}$. In other words, the out-transmission for node i is the sum of the ith column of M, and its in-transmission is the sum of the ith row of M.

	N_1	N_2		N_3	
Node (i)	$\mathrm{td}(i)$	$\sigma^+(i)$	$\sigma^-(i)$	$\sigma^+(i)$	$\sigma^-(i)$
1	8	15	12	∞	∞
2	8	14	9	8	∞
3	9	11	12	∞	∞
4	7	8	9	∞	∞
5	7	8	11	∞	∞
6	11	12	15	∞	11

Table 3. Total distance and transmission nodal measures for networks in Figure 4

Table 3 lists the total distance and transmission measures for the networks in Figure 4. The adjacency matrices were used in Algorithm 1 (Floyd-Warshall) using unit distances between adjacent nodes. Node 1 in network N_1 has a total distance of 8. This value is the sum of the distances from node 1 to each of the other nodes, $\mathrm{td}(i) = 1 + 2 + 2 + 1 + 2 = 8$, where each term in the sum is the shortest distance from node 1 to node 2, to node 3, and so forth.

2.2.1.4 Summary.

The nodal graph theoretic measures are repeated in Table 4. These measures indicate the connectedness (larger degree), centrality (smaller eccentricity), or status (larger total distance) of each node in the network [13, 12].

Table 4. Nodal measures from graph theory

Measure	Equation	Reference	Explanation
out-degree	$d^+(i) = \sum_{j \in V} a_{ij}$	[65:p. 58]	Indicate the extent to which a node is connected.
in-degree	$d^-(i) = \sum_{j \in V} a_{ji}$	[65:p. 58]	
degree	$d(i) = d^+(i) + d^-(i)$ (directed)	[65:p. 58]	
	$d(i) = \sum_{j \in V} a_{ij} = \sum_{j \in V} a_{ji}$ (undirected)	[65:p. 34]	
out-eccentricity	$e^+(i) = \max_{j \in V} d_N(i,j)$	[46:p. 381]	Indicate how central a node is within the network.
in-eccentricity	$e^-(i) = \max_{j \in V} d_N(j,i)$	[46:p. 381]	
eccentricity	$e(i) = \max(e^+(i), e^-(i))$ (directed)	[46:p. 381]	
	$e(i) = \max_{j \in V} d_N(i,j) = \max_{j \in V} d_N(j,i)$ (undirected)	[65:p. 71]	

Continued on next page

Table 4 – *Continued from previous page*

Measure	**Equation**	**Reference**	**Explanation**
total distance	$\mathrm{td}(i) = \sum_{j \in V} d_N(i,j)$	[13:p. 16]	
out-transmission	$\sigma^+(i) = \sum_{j \in V} d_N(i,j)$	[55:p. 1]	Indicate the status of a node.
in-transmission	$\sigma^-(i) = \sum_{j \in V} d_N(j,i)$	[55:p. 2]	

2.2.2 Network Measures - Graph Theory.

The remaining graph theoretic measures are representative of the entire network. The definitions of the nodal measures are used to compute some of the network measures. The measures presented in this section are a subset of the graph theoretic measures for entire networks. Each of the measures presented in this section may also be applied to sub-networks to indicate features of that portion of the underlying network structure.

2.2.2.1 Minimum, Maximum, and Average Degree.

A measure for the entire network related to the nodal degrees is the minimum degree $\delta(N)$. The minimum degree is the degree value of the node with the smallest degree. Similarly, the network has a minimum out-degree $\delta^+(N)$ and in-degree $\delta^-(N)$ measure [12, 65]

$$\delta(N) = \min_{i \in V} d(i), \tag{2.12}$$

$$\delta^+(N) = \min_{i \in V} d^+(i), \tag{2.13}$$

$$\delta^-(N) = \min_{i \in V} d^-(i). \tag{2.14}$$

The maximum degree $D(N)$ is the degree value of the node with the largest degree. Similarly, the network has a maximum out-degree $D^+(N)$ and in-degree $D^-(N)$ measure [12, 65],

$$D(N) = \max_{i \in V} d(i), \tag{2.15}$$

$$D^+(N) = \max_{i \in V} d^+(i), \tag{2.16}$$

$$D^-(N) = \max_{i \in V} d^-(i). \tag{2.17}$$

Network (N)	$\delta^+(N)$	$\delta^-(N)$	$\delta(N)$	$D^+(N)$	$D^-(N)$	$D(N)$	$\overline{d}(N)$
N_1	1	1	1	3	3	3	2.333
N_2	1	1	2	3	2	5	3
N_3	0	0	1	3	2	3	2.333

Table 5. Degree network measures for networks in Figure 4

The average degree of the network is [65]

$$\overline{d}(N) = \frac{\sum_{i \in V} d(i)}{n} = \frac{2m}{n}, \tag{2.18}$$

where n and m are the number of nodes and arcs, respectively. The average out- or in-degree is not calculated since $\sum_{i \in V} d^+(i) = \sum_{i \in V} d^-(i) = m$ [12], which makes the average out- or in-degree equivalent to half the average degree.

The degree measures (minimum, maximum, and average) are computed for the networks in Figure 4 using the measures in Table 1. The results are depicted in Table 5.

2.2.2.2 Diameter.

The diameter of a network $\text{diam}(N)$ is the longest of the (i, j)-shortest-paths, where $i \neq j$, for all possible node combinations. It is computed as the maximum of the nodal eccentricities [12, 65]

$$\text{diam}(N) = \max_{i \in V} e(i). \tag{2.19}$$

The diameter of a network can be computed from the output of Algorithm 1 as $\text{diam}(N) = \max_{i,j \in V} m_{ij}$. Recall that the eccentricity for a node in a directed network may be infinite; therefore, the diameter may also be infinite and can be a poor measure for directed networks. For flow networks, the diameter is infinite.

2.2.2.3 Radius.

The radius of a network $\text{rad}(N)$ is the shortest (i, j)-path for all possible node combinations. It is computed as the minimum of the nodal eccentricities

$$\text{rad}(N) = \min_{i \in V} e(i). \tag{2.20}$$

A relationship between the radius and diameter of an undirected, connected network is [13, 12]

$$\text{rad}(N) \leq \text{diam}(N) \leq 2\text{rad}(N). \tag{2.21}$$

However, it is not true, in general, for undirected networks that $\text{diam}(N) \leq 2\text{rad}(N)$ [12]. The other inequality, $\text{rad}(N) \leq \text{diam}(N)$, does hold.

The radius can be defined in terms of the out- and in-eccentricity measures. The *out-radius* is the minimum of the out-eccentricities and the *in-radius* is the minimum of the in-eccentricities [46],

$$\text{rad}^+(N) = \min_{i \in V} e^+(i), \tag{2.22}$$

$$\text{rad}^-(N) = \min_{i \in V} e^-(i). \tag{2.23}$$

Table 6 lists the diameter and radius measures for the networks in Figure 4. The radius and diameter of the networks are calculated using the eccentricity measures from Table 2. In network N_2, the diameter is larger than 2 times the radius, illustrating that (2.21) does not hold generally for directed networks [13, 12].

2.2.2.4 Center.

A *central node* of a network is a node that has the smallest eccentricity. Since the radius of the network is the smallest eccentricity value, the set of central nodes $C(N)$ of network

Network (N)	$\text{rad}^+(N)$	$\text{rad}^-(N)$	$\text{rad}(N)$	$\text{diam}(N)$
N_1	2	2	2	3
N_2	2	3	3	5
N_3	3	3	∞	∞

Table 6. Radius and diameter network measures for networks in Figure 4

N is [12]

$$C(N) = \{i | e(i) = \text{rad}(N)\} = \{i | e(i) = \min_{j \in V} e(j)\}. \tag{2.24}$$

The *center* of a network $\text{cen}(N)$ is the network induced by the central nodes, *i.e.* a network consisting of the central nodes and any arcs between them [12, 65].

The central nodes can be defined in terms of the out- and in-eccentricity measures as well. The *out-central nodes* are those nodes that have minimum out-eccentricities and the *in-central nodes* are those with minimum in-eccentricities [34, 46],

$$C^+(N) = \{i | e^+(i) = \text{rad}^+(N)\} = \{i | e^+(i) = \min_{j \in V} e^+(j)\}, \tag{2.25}$$

$$C^-(N) = \{i | e^-(i) = \text{rad}^-(N)\} = \{i | e^-(i) = \min_{j \in V} e^-(j)\}. \tag{2.26}$$

The *out-center* and *in-center* of a network are the networks induced by the out- or in-central nodes, respectively.

In Figure 4, the central nodes of network N_1 are nodes 1, 4, and 5, and are depicted in Figure 5. The node that is central and out-central in N_2 is node 4, while nodes 2 and 4 are in-central. In the flow network N_3, node 2, the source, is out-central and node 6, the terminus, is in-central. However, the center is all nodes since the radius is infinite for each node.

2.2.2.5 Periphery.

A *peripheral node* of a network is a node that has the largest eccentricity. Since the diameter of the network is the largest eccentricity, the set of peripheral nodes $P(N)$ of network N is [13, 12]

$$P(N) = \{i | e(i) = \text{diam}(N)\} = \{i | e(i) = \max_{j \in V} e(j)\}. \tag{2.27}$$

The *periphery* of a network $\text{per}(N)$ is the network induced by the peripheral nodes, *i.e.* a network consisting of the peripheral nodes and any arcs between them [13, 12].

Network N_1 of Figure 4 has a periphery consisting of nodes 2, 3, and 6, and are depicted in Figure 5. Nodes 1 and 6 are the peripheral nodes of N_2. All nodes are in the periphery of the flow network of N_3 since the diameter is infinite for all nodes.

2.2.2.6 Median.

A *medial node* of a network is the node with the minimum total distance [13]. The set of medial nodes $M(N)$ in network N is

$$M(N) = \{i | \text{td}(i) = \min_{j \in V} \text{td}(j)\}. \tag{2.28}$$

The *median* of a network $\text{med}(N)$ is the network induced by the medial nodes, *i.e.* a network consisting of the medial nodes and any arcs between them. Note that the median of a network need not be connected. The median is another concept dealing with the "middle" of the network [13].

The medial nodes can be defined in terms of the out- and in-transmission measures. The *out-medial nodes* are those nodes that have minimum out-transmissions and the *in-medial*

nodes are those with minimum in-transmissions,

$$M^+(N) = \{i | \sigma^+(i) = \min_{j \in V} \sigma^+(j)\}, \tag{2.29}$$

$$M^-(N) = \{i | \sigma^-(i) = \min_{j \in V} \sigma^-(j)\}. \tag{2.30}$$

The *out-median* and *in-median* of a network are the networks induced by the out- or in-medial nodes, respectively. The out-median of a flow network is the source (supersource) and the in-median is the terminus (superterminus).

The center, periphery, and median of network N_1 are depicted in Figure 5. Relative to the entire network, the central and medial nodes of network N_1 appear towards the middle of the network, while the peripheral nodes appear on the edges of the network. Note the subtle difference in the two measures indicating the "middle" of the network: the central nodes are nodes 1, 4, and 5 while the median consists of nodes 4 and 5. For N_2, the out-medial nodes are nodes 4 and 5 and the in-medial nodes are nodes 2 and 4. Here, the analyst must take care to determine which of the medial measures best provides insight into the problem being studied. In the flow network N_3, the out-medial node is node 2, the source, and the in-medial node is node 6, the terminus. While the center and periphery of the flow network consisted of all the nodes, the median is the source, node 2, since it is determined as the set of nodes with the minimum total distance from a node to all others. In addition, the total distance of all other nodes is infinite since there is no directed path from a non-source node to the source.

2.2.2.7 Wiener's Index and Average Distance.

Closely related to the median of a network is the *Wiener index* or the *transmission* of the network. The sum the shortest paths between all pairs of nodes in a network is the transmission of the network [55]. This quantity is also known as the Wiener index since

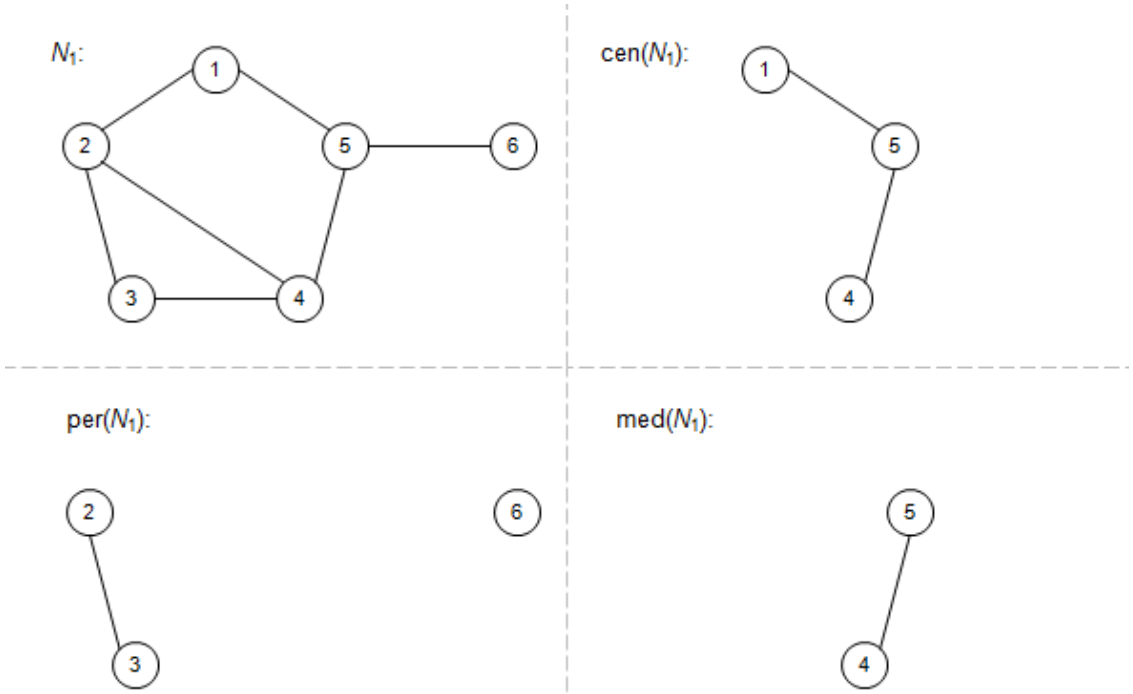


Figure 5. Center, periphery, and median of a network

Wiener used it to study the properties of paraffin's boiling point [66]. The Wiener index of a network $\mathrm{w}(N)$ is

$$\mathrm{w}(N) = \sum_{i,j \in V} d_N(i,j) = \sum_{i \in V} \mathrm{td}(i). \tag{2.31}$$

An equivalent definition of the Wiener index is sum of the total distances for each node in the network. In addition, the sum of out- or in-transmissions is equivalent to the Wiener index, $\mathrm{w}(N) = \sum_{i \in V} \sigma^+(i) = \sum_{i \in V} \sigma^-(i)$. As is the case for out- and in-transmissions, it is possible to attain an infinite index if there is no (i,j)-path in a directed network. The Wiener index for a flow network is infinite. The Wiener index of a network can be computed from the output of Algorithm 1 as $\mathrm{w}(N) = \sum_{i,j \in V} m_{ij}$.

The average distance in the network may be more intuitive for analysis. Since the average is the total of all distances divided by the possible number of pairs, the use of the Wiener index or average distance is equivalent. The average distance $\overline{\mathrm{w}}(N)$ of the network is [55, 65]

$$\overline{\mathrm{w}}(N) = \frac{\sum_{i,j \in V} d_N(i,j)}{\binom{n}{2}} = \frac{\mathrm{w}(N)}{n(n-1)}. \tag{2.32}$$

The Wiener index or average distance is minimized when the network is an undirected star and maximized when the network is an undirected path [65].

2.2.2.8 Eigenvalues and Connectivity.

The measures introduced in this section can be used to bound several of the measures described to this point and can be calculated using the output of Algorithm 1. When the networks become very large, the measures that use eigenvalues may be computed faster using linear algebra than those that rely on the algorithmic solutions.

The adjacency matrix A is the matrix representation of the connections within the network. The eigenvalues of A have interesting relationships to the other graph theoretic measures.

Let $n_{\text{eig}(A)}$ denote the number of distinct eigenvalues of A, then [65]

$$\text{diam}(N) < n_{\text{eig}(A)}. \tag{2.33}$$

The smallest and largest eigenvalues of A for network N are denoted $\lambda_{\min}(N)$ and $\lambda_{\max}(N)$, respectively. For a network N' induced by a subset of nodes in N [65],

$$\lambda_{\min}(N) \leq \lambda_{\min}(N') \leq \lambda_{\max}(N') \leq \lambda_{\max}(N). \tag{2.34}$$

This relationship extends to the the extreme eigenvalues when a node is removed from the network for which N is the original network and N' is the network with a node deleted [65].

The minimum, maximum, and average degree of a network are related to $\lambda_{\max}(N)$ in the following manner [65],

$$\delta(N) \leq \frac{2m}{n} \leq \lambda_{\max}(N) \leq D(N). \tag{2.35}$$

Another matrix representation of a network N is the Laplacian matrix of the network $L(N)$. Let $Q(N)$ denote the diagonal matrix whose entries are the degree of the nodes; in other words, the diagonal entries are $q_{ii} = d(i)$. In addition, the adjacency matrix is denoted by $A(N)$. Then the Laplacian matrix is $L(N) = Q(N) - A(N)$. For undirected networks, which have symmetric Laplacian matrices, $Q(N)$ has only real eigenvalues and its smallest eigenvalue is 0 [49]. The second smallest eigenvector is termed the *algebraic connectivity* of the network $a(N)$ [27].

There are interesting properties related to the algebraic connectivity of a network. For a network N with k nodes and their incident arcs removed, the relationship between the algebraic connectivity of the original network and the modified network N' is [27]

$$a(N') \geq a(N) - k. \tag{2.36}$$

The algebraic connectivity $a(N)$ is bounded by terms consisting of the minimum degree $\delta(N)$, the number of nodes n, and the number of arcs m is [27]

$$a(N) \leq \frac{n\delta(N)}{n-1} \leq \frac{2m}{n-1}. \tag{2.37}$$

$$a(N) \geq 2\delta(N) - n + 2. \tag{2.38}$$

For a network in which each of the n nodes is adjacent to every other node (a complete graph K_n in graph theory), the algebraic connectivity is the number of nodes in the network, $a(K_n) = n$ [27].

The *node connectivity* of a network $v(N)$ is the minimum number of nodes that must be removed from the network that result in the network being disconnected. The *arc connectivity* of a network $e(N)$ is the minimum number of arcs that must be removed from the network that result in the network being disconnected [65]. The node and arc connectivity

of a network are bounded above by the minimum degree of the network [65],

$$v(N) \leq e(N) \leq \delta(N). \tag{2.39}$$

When $m \geq n-1$, the node connectivity is bounded by the average degree of the network [12],

$$v(N) \leq \lfloor \overline{d}(N) \rfloor. \tag{2.40}$$

When the network has n nodes and is not complete, then the algebraic connectivity is bounded in terms of the node connectivity and arc connectivity as follows [27]

$$a(N) \leq v(N). \tag{2.41}$$

$$a(N) \geq 2e(N)\left(1 - \cos\left(\frac{\pi}{n}\right)\right). \tag{2.42}$$

The algebraic connectivity appears in the bounds for the diameter of the network [49]

$$\text{diam}(N) \geq \frac{4}{a(N)n}, \tag{2.43}$$

$$\text{diam}(N) \leq 2\left\lceil \frac{D(N) + a(N)}{4a(N)} \ln(n-1) \right\rceil. \tag{2.44}$$

The algebraic connectivity appears in bounds for the average distance $\overline{\text{w}}(N)$ for the network [49]

$$\overline{\text{w}}(N) \geq \frac{1}{n-1}\left(\frac{2}{a(N)} + \frac{n-2}{2}\right), \tag{2.45}$$

$$\overline{\text{w}}(N) \leq \frac{n}{n-1}\left\lceil \frac{D(N) + a(N)}{4a(N)} \ln(n-1) \right\rceil. \tag{2.46}$$

For the networks in Figure 4 and using unit distances for each arc, the Wiener index, average distance, number of distinct eigenvalues, and algebraic connectivity measures are

Network (N)	$\mathrm{w}(N)$	$\overline{\mathrm{w}}(N)$	$n_{\mathrm{eig}(A)}$	$a(N)$
N_1	50	1.667	6	0.722
N_2	68	2.267	4	1
N_3	∞	∞	1	1

Table 7. Several network measures for networks in Figure 4

depicted in Table 7. For the flow network N_3, the total distance of each non-source (terminus) node is infinite, and therefore, the Wiener index and average distance are also infinite.

2.2.2.9 Summary.

The network graph theoretic measures are summarized in Table 8. The degree-related network measures (minimum, maximum, and average degree) indicate the level of the network's activeness and can be a proxy for the central tendencies of the network. That is, to what extent the network (or sub-network) can be considered central or important. The remaining measures are related to distance (diameter, radius, central nodes, peripheral nodes, medial nodes, Wiener index, and average distance) indicate the relative size of the network in terms of the shortest path and the "centerness" of the network. Each of these measures will be examined to evaluate whether it gives an indication of the benefit to interdiction operations within the network. The social network analysis measures expand on this idea of central nodes in a network.

Table 8. Network measures from graph theory

Measure	Equation	Reference	Explanation
minimum degree	$\delta(N) = \min_{i \in V} d(i)$	[65:p. 34]	
minimum out-degree	$\delta^+(N) = \min_{i \in V} d^+(i)$	[65:p. 58]	Indicate the level of the network's activeness and can be a proxy for the central tendencies of the network.
minimum in-degree	$\delta^-(N) = \min_{i \in V} d^-(i)$	[65:p. 58]	
maximum degree	$D(N) = \max_{i \in V} d(i)$	[65:p. 34]	
maximum out-degree	$D^+(N) = \max_{i \in V} d^+(i)$	[65:p. 58]	
maximum in-degree	$D^-(N) = \max_{i \in V} d^-(i)$	[65:p. 58]	
average degree	$\overline{d}(N) = \frac{\sum_{i \in V} d(i)}{n} = \frac{2m}{n}$	[65:p. 35]	

Continued on next page

Table 8 – *Continued from previous page*

Measure	**Equation**	**Reference**	**Explanation**
central nodes	$C(N) = \{i \mid e(i) = \mathrm{rad}(N)\}$	[12] [65]	Represent the "center" of the network.
out-central nodes	$C^+(N) = \{i \mid e^+(i) = \mathrm{rad}^+(N)\}$	[34] [46:p. 381]	
in-central nodes	$C^-(N) = \{i \mid e^-(i) = \mathrm{rad}^-(N)\}$	[34] [46:p. 381]	
medial nodes	$M(N) = \{i \mid \mathrm{td}(i) = \min_{j \in V} \mathrm{td}(j)\}$	[13:p. 16]	
out-medial nodes	$M^+(N) = \{i \mid \sigma^+(i) = \min_{j \in V} \sigma^+(j)\}$	[13]	
in-medial nodes	$M^-(N) = \{i \mid \sigma^-(i) = \min_{j \in V} \sigma^-(j)\}$	[13]	

Continued on next page

Table 8 – *Continued from previous page*

Measure	**Equation**	**Reference**	**Explanation**
diameter	$\mathrm{diam}(N) = \max_{i \in V} e(i)$	[65:p. 71]	
radius	$\mathrm{rad}(N) = \min_{i \in V} e(i)$	[65:p. 71]	Indicate the relative size of the network in terms of the shortest path and the "centerness" of the network.
out-radius	$\mathrm{rad}^{+}(N) = \min_{i \in V} e^{+}(i)$	[46:p. 381]	
in-radius	$\mathrm{rad}^{-}(N) = \min_{i \in V} e^{-}(i)$	[46:p. 381]	
Wiener index	$\mathrm{w}(N) = \sum_{i,j \in V} d_N(i,j) = \sum_{i \in V} \mathrm{td}(i)$	[65:p. 72]	
transmission		[55]	
average distance	$\overline{\mathrm{w}}(N) = \dfrac{\mathrm{w}(N)}{n(n-1)}$	[65:p. 72]	
peripheral nodes	$P(N) = \{i \mid e(i) = \mathrm{diam}(N)\}$	[13:p. 16]	Represent the "edge" of the network.

2.2.3 Nodal Measures - SNA.

In social network analysis (SNA), several measures have been used to indicate the importance of nodes in a network. Some of these measures are the same as the graph theoretic measures, while others are similar. Typically in a SNA network, the nodes represent people and the arcs represent some connection (social, political, personal, and so forth) between them. The term "actors" is frequently used when referring to the nodes in a social network. The total distance of a node in graph theory is also termed the accessibility index [36] and is not reviewed again in this section.

2.2.3.1 Degree Centrality.

The degree centrality of a node, $C_D(i)$ in SNA literature, is equivalent to the graph theoretic measure degree given in (2.1). The degree centrality is standardized by dividing by the remaining number of nodes in the network $(n-1)$. This allows the comparison of the degree centrality of nodes in different sized networks [63]

$$C'_D(i) = \frac{d(i)}{n-1}. \tag{2.47}$$

The out-degree centrality $C_{D^+}(i)$ and in-degree centrality $C_{D^-}(i)$ of a node are equivalent to the graph theoretic measures out-degree (2.2) and in-degree (2.3) [53]. The degree centrality measures for directed networks are standardized in a similar manner,

$$C'_{D^+}(i) = \frac{C_{D^+}(i)}{n-1} = \frac{d^+(i)}{n-1}, \tag{2.48}$$

$$C'_{D^-}(i) = \frac{C_{D^-}(i)}{n-1} = \frac{d^-(i)}{n-1}. \tag{2.49}$$

$$C'_D(i) = \frac{C_{D^+}(i) + C_{D^-}(i)}{2(n-1)} = \frac{d^+(i) + d^-(i)}{2(n-1)} = \frac{C'_{D^+}(i) + C'_{D^-}(i)}{2}. \tag{2.50}$$

	N_1			N_2			N_3
Node (i)	$C'_D(i)$	$C'_{D^+}(i)$	$C'_{D^-}(i)$	$C'_D(i)$	$C'_{D^+}(i)$	$C'_{D^-}(i)$	$C'_D(i)$
1	0.4	0.2	0.2	0.2	0.2	0.2	0.2
2	0.6	0.2	0.4	0.3	0.6	0	0.3
3	0.4	0.2	0.2	0.2	0.2	0.2	0.2
4	0.6	0.4	0.4	0.4	0.2	0.4	0.3
5	0.6	0.6	0.4	0.5	0.2	0.4	0.3
6	0.2	0.2	0.2	0.2	0	0.2	0.1

Table 9. Standardized degree centrality measures for networks in Figure 4

Recall that the degree of a directed network is the sum of the out- and in-degrees (2.4). The standardized degree centrality for directed networks has $2(n-1)$ in the denominator since there are two times the number of other nodes in the network with which it can be connected (one for each direction). The standardized degree centralities measure the portion of the network with which a node has contact and represent the "potential communication activity" of a node [30].

The standardized degree centrality measures (out-degree, in-degree, and degree) are computed for the networks in Figure 4 using the measures in Table 1. The results are depicted in Table 9. In network N_2, the standardized out-degree centrality of node 4 is 0.4 since there are two arcs leaving the node out of the five possible remaining nodes to which arcs could lead. Likewise, the in-degree centrality and degree centrality for node 4 are 0.4 and 0.4, respectively.

2.2.3.2 Closeness Centrality.

The closeness centrality of a node $C_C(i)$ measures the centrality of a node based not only on its immediate neighbors, but also on the distance in the network from all other nodes, taking into account indirect links to others. Thus, it is related to the total distance measure of a node (2.9). The closeness centrality of node i and the standardized measure are [63:pp.

184–185]

$$C_C(i) = \frac{1}{\sum_{j \in V} d_N(i,j)} = \frac{1}{\mathrm{td}(i)}, \tag{2.51}$$

$$C'_C(i) = \frac{n-1}{\sum_{j \in V} d_N(i,j)} = (n-1)C_C(i). \tag{2.52}$$

Nodes that have no path between them, and therefore have an infinite total distance, would have a closeness measure of zero [53:p. 184].

The out- and in-transmissions can be used to compute the out-closeness centrality of a node $C_{C^+}(i)$ and its in-closeness centrality $C_{C^-}(i)$, respectively. These measures and their standardizations are

$$C_{C^+}(i) = \frac{1}{\sigma^+(i)}, \tag{2.53}$$

$$C'_{C^+}(i) = \frac{n-1}{\sigma^+(i)} = (n-1)C_{C^+}(i), \tag{2.54}$$

$$C_{C^-}(i) = \frac{1}{\sigma^-(i)}, \tag{2.55}$$

$$C'_{C^-}(i) = \frac{n-1}{\sigma^-(i)} = (n-1)C_{C^-}(i). \tag{2.56}$$

The out-closeness centrality of a node is also called radiality and measures the extent of the nodes reach into the network [61]. The in-closeness centrality of a node is also called integration and measures how well-connected a node is within the network [61]. Highly integrated nodes (those with the largest integration, or in-closeness scores) can be reached quickly [61].

Because the closeness centrality is related to the total distance and transmission, in flow networks with a single source and terminus, the source is the only node with non-zero out-closeness centrality and the terminus is the only node with non-zero in-closeness

	N_1		N_2				N_3			
Node (i)	$C_C(i)$	$C'_C(i)$	$C_{C^+}(i)$	$C'_{C^+}(i)$	$C_{C^-}(i)$	$C'_{C^-}(i)$	$C_{C^+}(i)$	$C'_{C^+}(i)$	$C_{C^-}(i)$	$C'_{C^-}(i)$
1	0.125	0.625	0.067	0.333	0.083	0.417	0	0	0	0
2	0.125	0.625	0.071	0.357	0.111	0.556	0.125	0.625	0	0
3	0.111	0.556	0.091	0.455	0.083	0.417	0	0	0	0
4	0.143	0.714	0.125	0.625	0.111	0.556	0	0	0	0
5	0.143	0.714	0.125	0.625	0.091	0.455	0	0	0	0
6	0.091	0.455	0.083	0.417	0.067	0.333	0	0	0.091	0.455

Table 10. Closeness centrality measures for nodes of networks in Figure 4

centrality. The closeness centralities measure the inverse average distance to all other nodes and represent the "independence or efficiency" of a node [30].

Table 10 lists the closeness centrality measures for the networks in Figure 4. The adjacency matrices were used in Algorithm 1 (Floyd-Warshall) with unit distances between adjacent nodes. As expected, the non-source and non-terminus nodes in the flow network (N_3) have out- and in-centrality values of 0, corresponding to their infinite out- and in-transmissions (Table 3).

2.2.3.3 Betweenness Centrality.

The betweenness centrality of a node $C_B(i)$ measures the extent to which nodes are along the shortest path between node pairs. These nodes that are between others may have more influence over the others since they appear on more shortest paths between the other nodes in the network. They have the potential to affect the flow of information along the path. The betweenness centrality of node k and its standardization are [10, 53, 63]

$$C_B(k) = \sum_{i \neq k \neq j \in V} \frac{n_{ij}(k)}{n_{ij}}, \tag{2.57}$$

$$C'_B(k) = \frac{1}{(n-1)(n-2)} \sum_{i \neq k \neq j \in V} \frac{n_{ij}(k)}{n_{ij}} = \frac{C_B(k)}{(n-1)(n-2)}, \tag{2.58}$$

where n is the number of nodes in the network, n_{ij} is the number of shortest (i, j)-paths, and $n_{ij}(k)$ is the number of shortest (i, j)-paths that include node k. The divisor in the

	N_1		N_2		N_3	
Node (k)	$C_B(k)$	$C'_B(k)$	$C_B(k)$	$C'_B(k)$	$C_B(k)$	$C'_B(k)$
1	2	0.10	2	0.10	1	0.05
2	3	0.15	7	0.35	0	0
3	0	0	7	0.35	0	0
4	6	0.30	11	0.55	3	0.15
5	9	0.45	11	0.55	4	0.20
6	0	0	0	0	0	0

Table 11. Betweenness centrality measures for nodes of networks in Figure 4

standardizing equation (2.58) is the total number of node pairs in the network, not including the node itself. [63]. The standardized betweenness is independent of arc length since it is based on the number of arcs on a shortest path and the total number of arcs in the network. The maximum standardized betweenness centrality is 1, which occurs for the central node of a star network [53]. The betweenness centrality measures can be computed whether the network is connected or directed [63]. The betweenness centrality values can also be computed whether the arc distances have a length of 1 or not [10].

The betweenness centrality is readily computed using the algorithm given by Brandes [10]. The algorithm runs in $O(nm)$ time for networks with unweighted arcs and $O(nm + n^2 \log n)$ time for weighted networks. Algorithm 2 details the unweighted version of the algorithm to compute betweenness centrality. The algorithm uses a shortest path discovery and counting routine (breadth first search) for each node and then accumulates the betweenness centrality using node dependencies. Node dependencies are computed using a recursive relation (used in the accumulation phase of the algorithm) so that n_{ij} and $n_{ij}(k)$ in Equation (2.57) are not directly computed. The algorithm presented is as given in another paper by Brandes [11].

Table 11 lists the betweenness centrality measure for the networks in Figure 4. Node 6 is not on any shortest paths in any of the three networks and has a betweenness centrality of 0 because the endpoints of the path are not included in the count.

Algorithm 2 Betweenness Centrality Algorithm (Unweighted Networks) [11:p. 5]

Input

A directed network with nodes $N = 1, \ldots, n$ and arcs $E = (i, j), i, j \in N$.

Data

Q: queue for nodes. S: stack for nodes. dist(v): distance from source. Pred(w): list of predecessors on shortest paths from source. $\sigma(v)$: number of shortest paths from source to node v. $\delta(v)$: dependency of source on node v.

Output

The betweenness centrality $C_B(i)$ for each node i.

Algorithm

For $s \in N$,
  Single-Source Shortest Paths Problem
    Initialization
      For $w \in N$, Pred$(w) \leftarrow$ empty list
      For $t \in N$, dist$(t) \leftarrow \infty$; $\sigma(t) \leftarrow 0$
      dist$(s) \leftarrow 0$; $\sigma(s) \leftarrow 1$
      Enqueue $s \rightarrow Q$
    While Q not empty
      Dequeue $v \leftarrow Q$; Push $v \rightarrow S$
      For each w such that $(v, w) \in E$
        Path Discovery
          if dist$(w) = \infty$
            dist$(w) \leftarrow$ dist$(v) + 1$
            Enqueue $w \rightarrow Q$
        Path Counting
          if dist$(w) =$ dist$(v) + 1$
            $\sigma(w) \leftarrow \sigma(w) + \sigma(v)$
            Append $v \rightarrow$ Pred(w)
  Accumulation
    For $v \in N$, $\delta(v) \leftarrow 0$
    While S not empty
      Pop $w \leftarrow S$
      For $v \in$ Pred(w)
        $\delta(v) \leftarrow \delta(v) + \frac{\sigma(v)}{\sigma(w)}(1 + \delta(w))$
      If $w \neq s$, $C_B(w) \leftarrow C_B(w) + \delta(w)$
Next s.

2.2.3.4 Clustering Coefficient.

The *clustering coefficient* is a measure of how many neighbors of a node are neighbors themselves [64]. This measure can be thought of as a localized version of betweenness, indicating the control a node has over its immediate neighbors [53]. Let $n_N(i)$ denote the number of neighbors of node i and $n_{CN}(i)$ be the number of the neighbors of node i that are connected. Then the clustering coefficient $C_{cl}(i)$ for node i is the ratio of the number of neighbors that are connected to the number of possible pairs of neighbors, [53, 64]

$$C_{cl}(i) = \frac{2n_{CN}(i)}{n_N(i)(n_N(i)-1)}, \tag{2.59}$$

for undirected networks and

$$C_{cl}(i) = \frac{n_{CN}(i)}{n_N(i)(n_N(i)-1)}, \tag{2.60}$$

for directed networks. The clustering coefficient is not standardized since it is already bounded between 0 and 1. The clustering coefficient is 0 when a node has only one neighbor or there are no neighbors that are connected. The value is 1 when every neighbor of a node i is connected to every other neighbor of node i.

The clustering coefficient for each node in the networks in Figure 4 are listed in Table 12. Since nodes 1, 5, and 6 have neighbors with no connections between them in each of the networks, the clustering coefficient of each of these nodes is 0 in all three example networks.

2.2.3.5 Summary.

The nodal SNA measures are summarized in Table 13. The degree measures (out-, in-, and degree centrality) indicate the amount of activity of a node in the network. The independence of a node is indicated by the closeness measures (out-, in-, and closeness centrality).

Node (i)	N_1 $C_{cl}(i)$	N_2 $C_{cl}(i)$	N_3 $C_{cl}(i)$
1	0	0	0
2	0.333	0.167	0.167
3	1	0.5	0.5
4	0.333	0.167	0.167
5	0	0	0
6	0	0	0

Table 12. Clustering coefficient measures for nodes of networks in Figure 4

The distance-type measures (betweenness centrality, out- and in-information centrality, and clustering coefficient) indicate the potential control a node has in the network. Finally, an indicator of the influence of a node is given by the eigen-type measures (eigenvector and in-eigenvector centrality, Katz centrality, PageRank and in-PageRank).

Table 13. Nodal measures from SNA

Measure	Equation	Reference	Explanation
degree centrality	$C_D(i) = d(i)$	[63:pp. 178-179]	
	$C'_D(i) = \frac{d(i)}{n-1}$		
out-degree centrality	$C_{D^+}(i) = d^+(i)$	[53:p. 169]	Indicate the amount of activity of a node in the network.
	$C'_{D^+}(i) = \frac{C_{D^+}(i)}{n-1} = \frac{d^+(i)}{n-1}$		
in-degree centrality	$C_{D^-}(i) = d^-(i)$	[53:p. 169]	
	$C'_{D^-}(i) = \frac{C_{D^-}(i)}{n-1} = \frac{d^-(i)}{n-1}$		
betweenness centrality	$C_B(k) = \sum_{i \neq k \neq j \in V} \frac{n_{ij}(k)}{n_{ij}}$	[10:p. 3]	Indicate the control a node has in the network.
	$C'_B(k) = \frac{C_B(k)}{(n-1)(n-2)}$	[53:p. 190]	

Continued on next page

Table 13 – *Continued from previous page*

Measure	**Equation**	**Reference**	**Explanation**
closeness centrality	$C_C(i) = \frac{1}{\sum_{j \in V} d_N(i,j)} = \frac{1}{\mathrm{td}(i)}$	[63:pp. 184–185]	
	$C'_C(i) = (n-1)C_C(i)$		
out-closeness centrality	$C_{C^+}(i) = \frac{1}{\sigma^+(i)}$	[61:p. 92]	Indicate the independence of a node.
radiality	$C'_{C^+}(i) = (n-1)C_{C^+}(i)$		
in-closeness centrality	$C_{C^-}(i) = \frac{1}{\sigma^-(i)}$	[61:p. 92]	
integration	$C'_{C^-}(i) = (n-1)C_{C^-}(i)$		

2.2.4 Network Measures - SNA.

There are several network measures from SNA that are useful. Some are the same as those reviewed in the graph theory section and are not repeated. Average degree is the same; the Wiener index is also termed the dispersion index within the SNA literature [36].

2.2.4.1 Beta Index.

The *beta index* of a network is the ratio of the number of nodes n in the network to the number of edges m. The beta index measures a network's complexity [36]. The beta index is

$$\beta(N) = \frac{n}{m}. \tag{2.61}$$

A planar network (a network in which intersecting arcs require a node) can not have a beta index larger than 3, while non-planar networks can have the beta index approach infinity [36].

2.2.4.2 Alpha Index.

The *alpha index* of a network is the ratio of the number of cycles in the network to the maximum number of cycles possible. The alpha index measures a network's redundancy [36]. The alpha index of a connected network is

$$\alpha(N) = \frac{m - n + 1}{\frac{n(n-1)}{2} - (n-1)}. \tag{2.62}$$

An alpha index of 0 indicates that the network has a tree structure and the removal a single arc would disconnect the network, while a value of 1 indicates that the network is completely connected (every pair of nodes is connected) [36].

2.2.4.3 Density / Gamma Index.

The *density* or *gamma index* of a network is the ratio of the number of arcs in the network to the maximum number of arcs possible. The density or gamma index measures a network's interconnection or availability of alternate routes [36, 63]. Because a directed network depends on the order of the node pairs, there are twice as many possible arcs. The gamma index for an undirected and a directed network are [63]

$$gamma_u(N) = \frac{2m}{n(n-1)} = \frac{\overline{d}}{n-1}, \tag{2.63}$$

$$gamma_d(N) = \frac{m}{n(n-1)}. \tag{2.64}$$

The gamma index indicates the saturation of the network in terms of arcs; a value of 0 indicates there are no arcs and a value of 1 indicates that all possible arcs are included in the network (all node pairs are connected) [36, 63].

2.2.4.4 Average Clustering Coefficient.

The *average clustering coefficient* is the average of the clustering coefficient across all nodes [53],

$$\overline{C_{cl}}(N) = \frac{\sum_{i \in V} C_{cl}(i)}{n}. \tag{2.65}$$

Average clustering coefficient measures the "cliquishness" of a group of nodes [64].

Each of the SNA network measures for the networks in Figure 4 is given in Table 14. The gamma index for N_1 is computed using (2.63), while the density for N_2 and N_3 is computed using (2.64). These measures do not have any special treatment (other than the differential between directed and undirected networks) for flow networks as some of the graph theoretic measures had.

Network (N)	$\beta(N)$	$\alpha(N)$	$\gamma(N)$	$\overline{C_{cl}}(N)$
N_1	1.167	0.200	0.467	0.278
N_2	1.500	0.400	0.300	0.139
N_3	1.167	0.200	0.233	0.139

Table 14. SNA network measures for networks in Figure 4

2.2.4.5 Summary.

The network SNA measures are repeated in Table 15. The complexity, redundancy, interconnectedness, and cliquishness of a network are indicated by the beta index, alpha index, density and average clustering coefficient, respectively.

Table 15. Network measures from SNA

Measure	Equation	Reference	Explanation
beta index	$\beta(N) = \frac{n}{m}$	[36:p. 120]	Indicates the complexity of the network.
alpha index	$\alpha(N) = \frac{m-n+1}{\frac{n(n-1)}{2}-(n-1)}$	[36:p. 120]	Indicates the redundancy of the network.
density	$\gamma_u(N) = \frac{2m}{n(n-1)} = \frac{\overline{d}}{n-1}$ (undirected)	[36:p. 120]	Indicates the interconnectedness of the network.
gamma index	$\gamma_d(N) = \frac{m}{n(n-1)}$ (directed)	[63:p. 101,129]	
average clustering coefficient	$\overline{C_{cl}}(N) = \frac{\sum_{i \in V} C_{cl}(i)}{n}$	[53:p. 204]	Indicates the cliquishness of the network [64].

2.3 Optimization Models

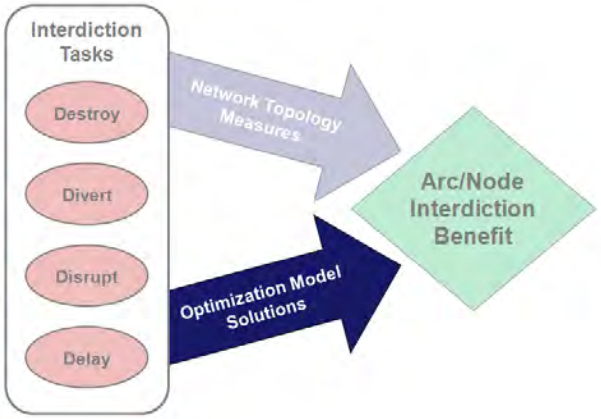


Figure 6. Research Framework: Optimization Model Literature

This section summarizes pertinent literature in the fields of network interdiction, providing a foundation for the models approach of the research framework, as depicted in Figure 6. Furthermore, this section reviews a number of papers that apply interdiction tasks in network interdiction models.

2.3.1 Network Interdiction.

Network interdiction has been studied in many application areas.

> The study of network interdiction began with military applications: disruption of the flow of enemy troops. More recent applications include infectious disease control, counter-terrorism, interception of contraband and illegal items such as drugs, weapons, or nuclear material, and the monitoring of computer networks [15:p. 1].

A common approach to modeling network interdiction is to formulate the problem in terms of a two-stage strategic game between two actors acting sequentially: the first seeking to disrupt, destroy, divert, or delay the capabilities of a network, and the second seeking to utilize the remaining network.

The bilevel programming problem (BLPP) is the mathematical formulation of a non-cooperative game between two players who each take a turn in a game in which each player has all information about the problem. In the economic literature, this is denoted as a Stackelberg game. The mathematical formulation [5:p. 6] of the BLPP is

$$\begin{aligned} \min_{x \in X} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leqq 0, \\ & \min_{y \in Y} \quad f(x, y), \\ & \text{s.t.} \quad g(x, y) \leqq 0. \end{aligned} \tag{2.66}$$

As can be seen in Equation (2.66), there is an optimization problem within the constraints. The second optimization problem, $\min_{y \in Y} f(x, y)$, is typically called the lower-level, inner, or follower's problem. In this research these terms are used interchangeably. The first optimization problem, $\min_{x \in X} F(x, y)$, called the upper-level, outer, or leader's problem, and is a minimization problem in the familiar notation of a mathematical programming problem. If the follower's problem has multiple optimal solutions, there will be uncertainty as to the definition of the minimization meaning. In other words, traditional minimization is valid only when the lower-level problem is uniquely determined. [22:p. 2] Additional constraints can be added to the formulation to improve realism.

The inherent hierarchy in the game implies perfect information; the decision of the leader will affect how the follower decides. Consider a simple production facility. Suppose the facility purchases material from one out of several choices of producers to make a single product that can be manufactured on one of a set of several machines. The leader in the BLPP that models the overall plant costs would be the management, which seeks to minimize the cost of material. The follower, based on the material purchased by management, makes the decision as to which machine to utilize to minimize the operating costs for manufacturing

the item. This simple example illustrates the hierarchical structure that can be modeled using bilevel programming. Based on this hierarchical structure, the leader-follower order is important and logical: switching the order would not make sense. The leader makes the optimal decision with the assumption that he or she has perfect knowledge of the follower's process. In other words, he or she will take into consideration the possible costs incurred based on the decision. An important characteristic of this type of problems is that the follower has autonomy and makes his own decisions taking into account the leader's decision. Thus, bilevel programming can be applied to a wide variety of applications that involve a hierarchical relationship. Some areas in which this arises are in transportation planning, price planning, network interdiction and infrastructure defense.

Wood [68] formulated a bilevel program to determine the arcs that should be destroyed to minimize the maximal flow through a network. The leader (attacker) seeks to cut arcs with the greatest impact to the network flow subject to a resource constraint, while the follower (defender) seeks to maximize the flow across the network. The problem formulation presented by Wood is slightly modified by explicitly listing the leader problem as an optimization problem to more clearly illustrate the bilevel structure. The objectives and constraints are unchanged. This model formulation, the bilevel maximum flow network interdiction model

(BMFNI), is: [68:p. 7]

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} \gamma_{ij} && && (2.67)\\
\text{s.t.} \quad & \sum_{(i,j)\in A} r_{ij}\gamma_{ij} \leq R && \\
& \gamma_{ij} \in \{0,1\}, && \forall (i,j) \in A \\
& \max \quad x_{ts} && \\
& \text{s.t.} \quad \sum_j x_{ij} - \sum_j x_{ji} = 0, && \forall (i,j) \in A \\
& \qquad \sum_j x_{sj} - \sum_j x_{js} - x_{ts} = 0 && \\
& \qquad \sum_j x_{tj} - \sum_j x_{jt} + x_{ts} = 0 && \\
& \qquad x_{ij} - u_{ij}(1-\gamma_{ij}) \leq 0, && \forall (i,j) \in A \\
& \qquad x_{ij} \geq 0, && \forall (i,j) \in A \cup (t,s),
\end{aligned}
$$

where γ_{ij} indicates whether the arc from node i to node j is interdicted, r_{ij} is the resource requirement to interdict the arc from node i to node j, R is the total amount of interdiction resources available, x_{ij} is the flow on the arc from node i to node j, A is the set of directed arcs, and u_{ij} is an upper bound on the flow (capacity) from node i to node j. In this formulation, only arcs are eligible to be cut.

To solve the problem, Wood reformulated the follower (max-flow) problem using its dual with the decision variable for the interdiction of arcs, γ_{ij}, fixed. The resulting model is linearized and the product of the dual variable and the $(1-\gamma_{ij})$ term replaced with β_{ij}. This allows certain constraints to be eliminated while others become equalities. The resulting model is an s-t cut model in which α_i indicates whether node i is on the s side of the cut ($\alpha_i = 0$) or the t side ($\alpha_i = 1$). Thus, an equivalent to the BMFNI model is the arc-only

network interdiction (AONI) model: [68:p. 6]

$$\min \quad \sum_{(i,j)\in A} u_{ij}\beta_{ij} \tag{2.68a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A} r_{ij}\gamma_{ij} \leq R \tag{2.68b}$$

$$\alpha_i - \alpha_j + \beta_{ij} + \gamma_{ij} \geq 0, \qquad \forall (i,j) \in A \tag{2.68c}$$

$$\alpha_s = 0, \tag{2.68d}$$

$$\alpha_t = 1, \tag{2.68e}$$

$$\alpha_i \in \{0,1\}, \qquad \forall i \in N \tag{2.68f}$$

$$\beta_{ij}, \gamma_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A. \tag{2.68g}$$

The value of β_{ij} indicates whether the arc from node i to node j is in the arc cut set but is not destroyed, *i.e.* flow remains uninterrupted. When $\beta_{ij} = 1$, arc (i,j) is in the cut set, but flow continues along the arc. When the value of γ_{ij} is 1, the arc from node i to node j is destroyed, i.e. there is no flow along the arc. All other β_{ij} and γ_{ij} are 0. Thus, the model identifies a cut set and selects arcs to destroy leaving the smallest remaining capacity possible given the resource constraints. The objective (2.68a) minimizes (attacker/leader's goal) the maximum (operator/follower's goal) flow across the network after the interdiction occurs. Constraint (2.68b) ensures that the follower does not exceed the resources available for indicting arcs. Constraint (2.68c) ensures that any arcs along the cut set are interdicted at most once. The source nodes are forced to the s side of the cut by Constraint (2.68d), while the sink nodes are forced to the t-side by Constraint (2.68e). Finally, the decision variables α, β and γ are all binary as indicated in Constraints (2.68f) and (2.68g).

Finally, Wood proved that the network interdiction problem where the attacker seeks to disrupt the (defender's) maximal flow through a capacitated network is NP-hard. The basic model developed is general so further variants of the problem can be examined [68].

Each of the network interdiction methods presented deal with the interdiction of arcs in the network. Kennedy *et al.* [45] develop a BLPP that model network interdiction from the standpoint of interdicting nodes rather than arcs. The leader is the network operator (or defender) who seeks to maximize flow through a capacitated network. The follower is the attacker whose objective is to remove nodes (and their outgoing arcs), so as to minimize the maximal s-t flow. The solution approach was to reformulate the follower problem so all arcs flowing out of targeted nodes are destroyed and transform the entire problem to a mixed integer problem. The node interdiction model was tested on randomly generated networks to determine the improvement in interdicting arcs rather than using the traditional node splitting technique to create dummy arcs. Finally, the node interdiction model solved a realistic communications network interdiction problem.

The network interdiction models reviewed in this section present the foundational framework that is extended in the literature. There have been numerous works extending this framework, however, the network interdiction foundation with the goal of destroying or disrupting arcs or nodes is established with the literature presented.

2.3.2 Network Diversion.

Within the literature, there are a number of papers that address the network *diversion* problem. However, these models determine the interdiction benefit of nodes and arcs to channeling by destroying arcs, rather than diverting as defined in joint interdiction doctrine. Because the terms used are similar, this section presents a detailed and comprehensive review of network diversion.

The network diversion problem is a type of network interdiction problem where the leader attacks a network by destroying arcs to channel flow to at least one member of a set of predetermined arcs. The network diversion problem was first posed in the open literature in 2001 by Curet [20]. The goal of the problem is to produce a minimum source-terminus cut so that any remaining source-terminus paths includes at least one arc that is included in the *diversion set*, a specified subset of arcs. Consider a directed graph $G(N, E)$ where N denotes the set of nodes and E denotes the set of edges (arcs). The network diversion problem is formulated as an integer programming formulation with three types of binary decision variables: z_{ij} indicates whether arc (i, j) is included in the cut ($z_{ij} = 1$, 0 otherwise), y_i indicates whether node i is on the source side of the cut ($y_i = 0$) or the terminus side ($y_i = 1$), and x_{ij} indicates whether flow traverses arc (i, j) ($x_{ij} = 1$, 0 otherwise). The diversion set is denoted D and consists of arcs only. Let $\overline{D}$ denote the set of arcs in E but not in D. Finally, the parameter ε defines the importance of diverting relative to interdiction

costs. The network diversion formulation follows [20:p. 38]:

$$\min \quad \sum_{(i,j)\in\overline{D}} (c_{ij} z_{ij}) + \varepsilon \sum_{(i,j)\in\overline{D}} x_{ij} \tag{2.69a}$$

$$\text{s.t.} \quad y_i - y_j + z_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{2.69b}$$

$$y_i \in \{0,1\}, \qquad \forall i \in N, \tag{2.69c}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{2.69d}$$

$$y_t = 1, y_s = 0, \tag{2.69e}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{k:(k,i)\in E} x_{ki} = b_i, \qquad \forall i \in N, \tag{2.69f}$$

$$x_{ij} \in 0,1, \qquad \forall (i,j) \in E, \tag{2.69g}$$

$$\sum_{(i,j)\in D} z_{ij} \geq 1, \tag{2.69h}$$

$$x_{ij} + z_{ij} \leq 1, \qquad \forall (i,j) \in \overline{D}, . \tag{2.69i}$$

The objective function (2.69a) has two terms: the first minimizes the cut set cost and the second weights the importance of flow diversion along the shortest path through the diversion set. Constraints (2.69b)-(2.69e) are the formulation to determine a minimum cut, which is the dual of the maximum flow problem [6:p. 598]. Constraints (2.69f)-(2.69g) along with defining b_i to conserve flow ($b_s = 1, b_t = 1, b_i = 0, \forall i \neq s, t$) ensure a single source-terminus path exists having a flow value of 1. Constraint (2.69h) forces the selected cut set to include at least one arc in the diversion set. Finally, Constraint (2.69i) allows the model to use arcs in the diversion set for both cutting the source from the terminus and appearing on the source-terminus path.

Curet [20] then uses Lagrangian relaxation to solve the network diversion in real time. The Lagrangian relaxation algorithm moves the linking constraint (2.69i) to the objective. The algorithm solves the Lagrangian problem with the weight of the linking constraint set

to 0. If the solution is such that the linking constraint is feasible, the solution is optimal. Otherwise, a subgradient (and step-length) is used to update the Lagrange multiplier. The algorithm iterates until an optimal solution is determined. The Lagrangian algorithm is implemented in an enumeration framework (determining all cut sets [21]) to solve the network diversion problem and the results show that the algorithm indeed provided solutions in real time.

Cintron-Arias *et al.* [14] used a similar model formulation except that they examined the case of a single arc in the diversion set (arc (v, w)). They formulated a heuristic approach to solve the network diversion problem. The heuristic determines the shortest s-v path and the shortest w-t path separately. The minimum cut across the network is then determined while disregarding arcs on these paths. Next, the heuristic iterates through each arc on the paths selected in the first step and removes the arc from inclusion in the shortest s-v or w-t path and updates the shortest path and associated minimum cut. The authors recommend considering shortest cardinality paths as well as paths with smallest cost-to-path-length ratios. The heuristic found the optimal solution in some test cases. The authors do not mention the time to complete the heuristic for the problems tested.

Erken [26] solved the network diversion problem using a branch-and-bound framework. The diversion arc (equivalent to the single arc (v, w) of Cintron-Arias *et al.* [14]) is forced to be part of the minimum cut. If the cut identified is minimal, the solution is optimal for the diversion problem. Otherwise, branching occurs by including and excluding an arc in the cut set. The algorithm continues until the minimum cut set containing the diversion edge is located.

Cullenbine *et al.* [19] strengthen the formulation for a diversion set with a single edge (arc (v, w)) by taking advantage of the fact that the diversion edge must be on the source-terminus path and that the tail node of the diversion edge will be on the source side of the cut set and the head node will be on the terminus side of the cut set. The single-commodity

integer formulation using the same variable names as in (2.69a) follows [19:p. 10]:

$$\min \quad \sum_{(i,j)\in E} (c_{ij} z_{ij}) + \varepsilon \sum_{(i,j)\in E} (x_{ij}) \tag{2.70a}$$

$$\text{s.t.} \quad y_i - y_j + z_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{2.70b}$$

$$y_s = 0, y_t = 1, \tag{2.70c}$$

$$y_v = 0, y_w = 1, \tag{2.70d}$$

$$z_{vw} = 1, \tag{2.70e}$$

$$\sum_{j|(i,j)\in E} x_{ij} - \sum_{j|(j,i)\in E} x_{ji} = b_i, \qquad \forall i \in N, \tag{2.70f}$$

$$x_{vw} = 1, \tag{2.70g}$$

$$z_{ij} + z_{ji} + x_{ij} + x_{ji} \leq 1, \qquad \forall (i,j) \in E - (v,w) | i < j \text{ and } (j,i) \in E, \tag{2.70h}$$

$$z_{ij} + x_{ij} \leq 1, \qquad \forall (i,j) \in E - (v,w) | (j,i) \notin E, \tag{2.70i}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{2.70j}$$

$$y_i \in \{0,1\}, \qquad \forall i \in N, \tag{2.70k}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, . \tag{2.70l}$$

The decision variable, x_{ij}, is not restricted to be binary in this formulation. By limiting y_i and z_{ij} to binary values, the solutions of the LP relaxation result in binary values for x_{ij} values. Cullenbine *et al.* conducted extensive testing to verify the efficiency of this combination of binary and continuous variables [19:p. 10]. In addition to the flow conservation restrictions ($b_s = 1, b_t = 1, b_i = 0, \forall i \neq s,t$), the interdiction cost for the diversion arc is 0 ($c_{vw} = 0$). The objective (2.70a) and Constraints (2.70b)-(2.70c), (2.70f), and (2.70i)-(2.70l) are as described for (2.69a)-(2.69i). Constraints (2.70d), (2.70e), and (2.70g) are introduced since the diversion set is assumed to contain only one arc. Finally, Constraint (2.70h) is included to address antiparallel arcs (*i.e.*, those arcs that have flow in opposite directions between

the same pair of nodes). The authors' extensive testing indicates that requiring only the y_i decision variable to be binary while the x_{ij}- and y_i-variables can take non-zero positive real values. A value of $\varepsilon < \frac{1}{|N|}$ ensures the penalty is less than 1 and that the shortest path problem does not preempt the minimum cost cut.

Cullenbine *et al.* [19] then formulate a two-commodity formulation consisting of two subpaths. One commodity flows from the source to the tail node in the single diversion edge (arc (v, w)) along the first subpath, while the second commodity flows from the head node of the diversion arc to the terminus. The stronger formulation, which essentially finds a shortest s-v path, a shortest w-t path, and a minimum s-t cut that includes the diversion edge and no arcs along the shortest paths, follows [19:pp. 10-11]. The flow decision variables are updated to account for each of the commodity paths: x_{ij}^S indicates whether flow traverses arc (i, j) along the shortest s-v path ($x_{ij}^S = 1$, 0 otherwise) and x_{ij}^T indicates whether flow

traverses arc (i,j) along the shortest w-t path ($x_{ij}^T = 1$, 0 otherwise).

$$\min \quad \sum_{(i,j)\in E} (c_{ij} z_{ij}) + \varepsilon \sum_{(i,j)\in E} (x_{ij}^S + x_{ij}^T) \tag{2.71a}$$

$$\text{s.t.} \quad y_i - y_j + z_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{2.71b}$$

$$y_s = 0, y_t = 1, \tag{2.71c}$$

$$y_v = 0, y_w = 1, \tag{2.71d}$$

$$z_{vw} = 1, \tag{2.71e}$$

$$\sum_{j|(i,j)\in E} x_{ij}^S - \sum_{j|(j,i)\in E} x_{ji}^S = b_i^S, \qquad \forall i \in N, \tag{2.71f}$$

$$\sum_{j|(i,j)\in E} x_{ij}^T - \sum_{j|(j,i)\in E} x_{ji}^T = b_i^T, \qquad \forall i \in N, \tag{2.71g}$$

$$x_{vw}^S = 0, x_{vw}^T = 0, \tag{2.71h}$$

$$z_{ij} + z_{ji} + x_{ij}^S + x_{ji}^S + x_{ij}^T + x_{ji}^T \leq 1, \qquad \forall (i,j) \in E - (v,w) | i < j \text{ and } (j,i) \in E, \tag{2.71i}$$

$$z_{ij} + x_{ij}^S + x_{ij}^T \leq 1, \qquad \forall (i,j) \in E - (v,w) | (j,i) \notin E, \tag{2.71j}$$

$$x_{ij}^S, x_{ij}^T \geq 0, \qquad \forall (i,j) \in E, \tag{2.71k}$$

$$y_i \in \{0,1\}, \qquad \forall i \in N, \tag{2.71l}$$

$$z_{ij} \geq 0, \qquad \forall (i,j) \in E, . \tag{2.71m}$$

The explanations for the model are the same as those given for the model given in (2.70a)-(2.70l). Since the path from the source to the diversion arc is on the source-side of the cut set and the path from the diversion arc to the terminus is on the terminus-side of the cut

set, the following two inequalities will hold:

$$\sum_{j|(i,j)\in E} x_{ij}^S + y_i \leq 1, \qquad \forall i \in N - t, \tag{2.71n}$$

$$\sum_{j|(i,j)\in E} x_{ij}^T - y_i \leq 0, \qquad \forall i \in N - s, . \tag{2.71o}$$

Computational results indicate that the model, with these inequalities, solve problem instances an order of magnitude faster than the the original model (2.69a)-(2.69i). The authors conclude stating that the model (2.71a)-(2.71o) should be the standard network interdiction model based on its tight linear programming relaxation.

2.3.3 Network Disruption.

In practice, whenever an object is targeted, there is a chance that the attack was only partially successful. Diminishing the capacity of the network based on a probability of damage is more realistic than assuming the target is destroyed. One pertinent method addresses network interdiction when the targeted arcs are not destroyed, but rather disrupted.

Israeli and Wood [42] develop a BLPP to solve a shortest-path network interdiction problem. The problem is formulated as a BLPP where the leader (or attacker) increases the arc lengths of targeted arcs to maximize the network's shortest path, *i.e.* he or she seeks to delay the time of traversing the network. The follower is the network operator who directs flow through the shortest path on the network. Their formulation for maximizing the shortest path follows. The model is slightly modified to illustrate the bilevel structure and with arcs denoted as an (i, j) pair rather than using a single index k, as was used in the

original formulation [42:p. 99].

$$\max \quad x_{ij} \tag{2.72a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} r_{ij}x_{ij} \leq R, \tag{2.72b}$$

$$x_{ij} \in 0, 1, \qquad \forall (i,j) \in E, \tag{2.72c}$$

$$\min \quad \sum_{k} (c_{ij} + d_{ij}x_{ij})y_{ij} \tag{2.72d}$$

$$\text{s.t.} \quad \sum_{j} y_{ij} - \sum_{j} y_{ji} = 0, \qquad \forall i \in N - \{s,t\} \tag{2.72e}$$

$$\sum_{j} y_{sj} - \sum_{j} y_{js} = 1, \tag{2.72f}$$

$$\sum_{j} y_{tj} - \sum_{j} y_{jt} = -1, \tag{2.72g}$$

$$y_{ij} \geq 0, \qquad \forall (i,j) \in E. \tag{2.72h}$$

The decision variable of the leader is x_{ij} which indicates whether arc (i,j) is interdicted. The follower determines the shortest path over the network via the variable y_{ij} which takes a value of 1 if the arc is on the shortest path after interdiction and 0 otherwise. The follower's problem, consisting of (2.72d)-(2.72h) is the standard shortest path formulation after taking into account the new arc distances. The leader is limited to R units of resource for interdicting arcs, with the interdiction cost per arc denoted by r_{ij}. The problem is solved using Benders decomposition with supervalid inequalities for the master (upper) problem [42]. Examples show the improvement of solutions using the decomposition algorithm over the results in the problem using direct solution techniques.

2.4 Summary

This chapter reviewed literature pertinent to this research. First, a number of graph theoretic and social network analysis measures were reviewed that may offer insights into the interdiction benefit of network models. Finally, optimization models whose solutions identify critical nodes and/or arcs to network interdiction were examined. In the following chapters, the foundational work highlighted in this chapter will be extended.

III. Measures After Destroying Nodes

3.1 Introduction

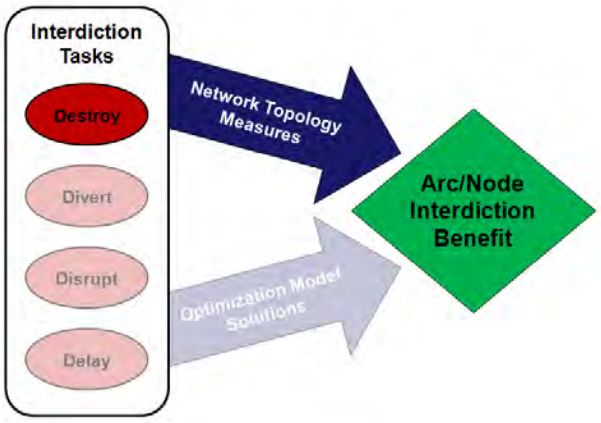


Figure 7. Research Framework: Measures After Destroying Nodes

The research framework is depicted in Figure 7 with portions faded to highlight the focus of this chapter. The measures approach begins with a network and uses nodal measures to assess the potential benefit of each for interdiction. In this chapter, a network algorithm is developed to readily compute and/or update measures when a node is destroyed. The focus is the 'destroy' interdiction task within the measures approach of the research framework, as depicted in Figure 7. In general, planners favor targeting nodes rather than arcs for interdiction operations against infrastructures. Within social networks, nodes are also more frequently targeted since nodes typically represent people and arcs, relationships. Therefore, in this chapter it is assumed that only nodes are targeted for destruction.

3.2 Geodesics and Related Measures

This section develops and demonstrates the utility of storing information about the geodesics, *i.e.*, shortest path lengths, between all node pairs in a network of interest to

enable more rapid calculation of selected node and network measures. Knowledge of the geodesics between all node pairs in a network is required to compute a number of measures that characterize the network. *A priori* processing of this information enables faster subsequent computations of measures that depend on the geodesics. Several algorithms exist that compute a matrix of geodesics for weighted and unweighted networks. Analysis of measure computations on randomly generated networks, both with and without the proposed stored information, are compared and contrasted to demonstrate the importance of preprocessing and retaining the geodesic information.

The reduction in calculation time via preprocessing the geodesics enables real-time network analysis. Consider a situation in which an enemy's social network is known *a priori* and a reconnaissance drone has been tracking a high-priority enemy personnel target, *e.g.*, terrorists, drug dealers, jihadists, or others. During the mission, the drone observes a group of additional known enemy personnel (whose whereabouts were unknown until the observed meeting) entering three vehicles. Unfortunately, the assets available to a decision-maker allow for the targeting of only one vehicle. (The term targeting refers to either lethal or nonlethal actions such as detailed observation, signals collection, or possibly destruction.) In the case when the distances in the social network represent weights, rapid analysis of the social network (for which the geodesic matrix is known) allow the decision maker to target the vehicle that yields the largest desired effect. It may be more beneficial to target one of the two vehicles not carrying the original primary high-value target. By computing the measures using the geodesic matrix, such rapid analysis enables target selection within a narrow time window for a required decision, *e.g.*, before an intelligence feed is lost or the vehicles disperse.

3.2.1 Nodal Measures Related to Geodesics.

For reference, consider a network N. The network consists of a set V of vertices or nodes in which each of the n nodes is indexed $i = 1, \ldots, n$. The set of arcs is denoted E and individual arcs are identified by pairs (i, j), where $i, j \in V$. Let G be the geodesic matrix with entries g_{ij} to denote the length of the geodesic from node i to node j in the network.

There exist several nodal and network measures for which calculations require geodesic information. A brief explanation of several of the measures that require such information follows. A subset of the measures are specific to individual nodes in a network. These nodal measures are listed in Table 16 along with the respective equation (in terms of the geodesic matrix G) for their computation and references from the literature. Likewise, network measures which are representative of the entire network and require the geodesic matrix are provided in Table 17.

The *eccentricity* $e(i)$ of node i is the maximum of the geodesics from node i to all other nodes. The eccentricity of i is the length of the shortest path from node i to the node that is the farthest distance away. For undirected, connected networks, each node will have a finite eccentricity [65:p. 71].

The *total distance* $\mathrm{td}(i)$ of node i in a network is the sum of the length of the geodesics from node i to all other nodes in the network [12]. In a social network, the total distance may indicate how quickly information from an individual represented by a node can reach all others, where it is assumed people with smaller total distance measures reach everyone more quickly. The total distance is sometimes referred to as the *status* of the node [13]. A node closest to all other nodes in the network has status.

The *closeness centrality* of a node $C_C(i)$ measures the centrality, *i.e.*, a measure of how central a node is within the network, of a node based not only on its immediate neighbors, but also on the distance in the network from all other nodes, thereby accounting for the geodesics to all non-adjacent nodes as well. It is computed as the inverse of the total distance measure

Measure	Equation	Reference
eccentricity	$e(i) = \max(\max_{j \in V} g_{ij}, \max_{j \in V} g_{ji})$ (directed)	[46:p. 381]
	$e(i) = \max_{j \in V} g_{ij} = \max_{j \in V} g_{ji}$ (undirected)	[65:p. 71]
total distance (transmission)	$\mathrm{td}(i) = \sum_{j \in V} g_{ij}$	[13:p. 16]
closeness centrality	$C_C(i) = \frac{1}{\sum_{j \in V} g_{ij}} = \frac{1}{\mathrm{td}(i)}$	[63:pp. 184–185]

Table 16. Nodal measures related to geodesics

of a node. The closeness centrality of a node is also called *radiality*, and it measures the extent of a node's reach into the network [61].

3.2.2 Network Measures Related to Geodesics.

The nodal measures are utilized to identify characteristics of every node. Network measures on the other hand, are utilized to identify characteristics of the entire network or selected nodes as they relate to the entire network.

The *diameter* of a network $\mathrm{diam}(N)$ is the longest of the (i, j)-geodesics, where $i \neq j$, for all possible node combinations. It is computed as the maximum of the nodal eccentricities of a network [12, 65].

The *radius* of a network $\mathrm{rad}(N)$ is the length of the minimum geodesic in the network [65:p. 71]. It is computed as the minimum of the nodal eccentricities.

A *peripheral node* of a network is a node that has the largest eccentricity. In other words, the peripheral nodes are those nodes with their eccentricity equal to the diameter of the network [13, 12].

A *central node* of a network is a node that has the smallest eccentricity, *i.e.*, its eccentricity is equal to the radius of the network.

Measure	Equation	Reference
diameter	$\mathrm{diam}(N) = \max_{i \in V} e(i)$	[65:p. 71]
radius	$\mathrm{rad}(N) = \min_{i \in V} e(i)$	[65:p. 71]
peripheral nodes	$P(N) = \{i \mid e(i) = \mathrm{diam}(N)\}$	[13:p. 16]
central nodes	$C(N) = \{i \mid e(i) = \mathrm{rad}(N)\}$	[12:p. 16] [65:p. 72]
medial nodes	$M(N) = \{i \mid \mathrm{td}(i) = \min_{j \in V} \mathrm{td}(j)\}$	[13:p. 16]
Wiener index (transmission)	$\mathrm{w}(N) = \sum_{i,j \in V} g_{ij} = \sum_{i \in V} \mathrm{td}(i)$	[49:p. 268] [55:p. 2] [65:p. 72]
average distance	$\overline{\mathrm{w}}(N) = \frac{\mathrm{w}(N)}{n(n-1)}$	[65:p. 72]

Table 17. Network measures related to geodesics

A *medial node* of a network is the node with the minimum total distance to all other nodes in the network [13].

Closely related to the medial nodes of a network is the *Wiener index* or the *transmission* of the network $\mathrm{w}(N)$. The sum the geodesics between all pairs of nodes in a network is the transmission of the network [55]. This quantity is also known as the Wiener index. (Harry Wiener showed that this quantity is correlated with paraffin's boiling point [66]). An equivalent definition of the Wiener index is sum of the total distances for each node in the network.

The *average distance* in the network $\overline{\mathrm{w}}(N)$ may be more intuitive for analysis. Since the average is the total of all distances divided by the possible number of pairs, the use of either the Wiener index or average distance is equivalent for a given network. However, the use of the average distance scales this network measure and enables the comparison of different sized networks.

3.2.3 All Geodesics Algorithms.

Implementing an algorithm to determine all geodesics for all node pairs makes the calculations of these measures tractable. There exist a number of algorithms for determining the geodesics between all node pairs in a network. Three such algorithms are considered: the Floyd-Warshall (F-W) Algorithm, the Breadth First Search (BFS) Algorithm, and a Repeated Dijkstra's (RD) Algorithm implemented using heaps.

In each of these algorithms, a geodesic matrix is computed, where the value of element (i, j) represents the geodesic length from node i to node j and is undefined (or set to equal to machine infinity for practical implementation) if there exists no (i, j) path. The term *geodesic matrix* is used to avoid confusion with the distance matrix D, where the value of element (i, j) represents the length of the arc from node i to node j. Accepted practices indicate that the F-W algorithm is appropriate for dense, weighted networks whereas the RD algorithm is more suited to sparse, weighted networks [8]. Meanwhile, the BFS algorithm is typically recommended for unweighted networks.

The Floyd-Warshall (F-W) Algorithm (denoted Algorithm 3) identifies the geodesic distance between all node combinations. The implementation presented is based on the algorithmic statement in Ahuja *et al.* [3:p. 148]. A contribution of the algorithm is its induction step that utilizes dynamic programming to iteratively compute the geodesic lengths. This insight is attributed to Warshall [62], whereas the algorithm's present form is attributed to Floyd [28]. The F-W Algorithm, as outlined in Algorithm 3, runs in $O(n^3)$ time because it implements n iterations, each of which conducts n^2 compare-and-update operations [3:p. 148]. The F-W algorithm analyzes both directed and undirected networks. The distance matrix utilized in the F-W Algorithm may be weighted or unweighted with no material impact on running time. To consider an unweighted distance matrix utilizing this algorithm, set the length of each arc to 1 or, equivalently, use the adjacency matrix as a proxy.

Algorithm 3 Floyd-Warshall (F-W) Algorithm [3:p. 148]

Input

Directed network with nodes $V = \{1, \ldots, n\}$ and arcs $E = (i, j), i, j \in V$.
A: matrix of adjacency between node pairs.
D: matrix of distances between node pairs.

Output

F: matrix of geodesics between all node pairs.

Algorithm

Initialization
For $i, j \in V$, $f_{ij} \leftarrow \infty$
For $i \in V$, $f_{ii} \leftarrow 0$
For $(i, j) \in E$, $f_{ij} \leftarrow d_{ij}$
Compute Geodesics
For $k \in V$,
For (i, j) in $V \times V$,
If $f_{ij} > f_{ik} + f_{kj}$
$f_{ij} \leftarrow f_{ik} + f_{kj}$.
Next (i, j).
Next k.

Alternatively, consider determining the lengths of all geodesics by implementing a Repeated Dijkstra's (RD) Algorithm, denoted Algorithm 4. The geodesic matrix is populated by implementing Dijkstra's Algorithm using each node as the source node, successively. Thus, each iteration computes the geodesic lengths for a given row in the geodesic matrix. The RD Algorithm as presented is based on the implementation in Ahuja *et al.* [3:p. 115] and exhibits a running time of $O(n^3)$. If implemented using a Fibonacci heap, the running time reduces to $O(n^2 \ln n)$ [18:p. 530]. However, in this study the RD Algorithm was implemented in MATLAB utilizing a priority queue. As in the F-W Algorithm, the distance matrix utilized in the RD Algorithm may be weighted or unweighted (*i.e.*, by letting $D = A$) with no material impact on running time.

Both of the aforementioned all pairs geodesic algorithms can be applied to unweighted networks. However, a more effective algorithm is available. The Breadth First Search (BFS) Algorithm, denoted Algorithm 5, examines the adjacent nodes of a searched node to find

Algorithm 4 Repeated Dijkstra's (RD) Algorithm [3:p. 115]

Input

Directed network with nodes $V = \{1, \ldots, n\}$ and arcs $E = (i, j), i, j \in V$.
A: matrix of adjacency between node pairs.
D: matrix of distances between node pairs.

Data

Q: priority queue for nodes.

Output

G: matrix of geodesics between all node pairs.

Algorithm

Initialization
For $s, t \in V$, $g_{st} \leftarrow \infty$
For $s \in V$,
Single-Source Geodesics Problem
$g_{ss} \leftarrow 0$
Enqueue $s \rightarrow Q$ with priority 0
While Q not empty
Extract minimum $v \leftarrow Q$
For each w such that $(v, w) \in E$
$d = g_{sv} + d_{vw}$
if $g_{sw} > d$
$g_{sw} \leftarrow d$
If $w \in Q$
Enqueue $w \rightarrow Q$ with priority d
Else
Decrease priority of $w \in Q$ to d
Next s.

possible geodesics. The BFS algorithm is very effective for unweighted networks because the arcs to each adjacent node are in the geodesic from the source. The implementation in Algorithm 5 is slightly modified from the presentation of Cormen *et al.* [18:p. 470]. Differences include utilizing a queue rather than coloring the nodes, implementing a loop for each node (as in the RD Algorithm) to obtain the geodesic matrix, and mirroring the notation employed in Algorithms 3 and 4. The running time of the BFS Algorithm for all nodes is $O(n^2 + nm)$, where m is the number of arcs in the network [18:p. 470]. The terms n^2

Algorithm 5 Breadth First Search (BFS) Algorithm [18:p. 470]

Input

Directed network with nodes $V = \{1, \ldots, n\}$ and arcs $E = (i, j), i, j \in V$.
A: matrix of adjacency between node pairs.

Data

Q: queue (FIFO) for nodes.

Output

H: matrix of geodesics between all node pairs.

Algorithm

Initialization
For $s, t \in V$, $h_{st} \leftarrow \infty$
For $s \in V$,
Single-Source Geodesics Problem
$h_{ss} \leftarrow 0$
Enqueue $s \rightarrow Q$
While Q not empty
Dequeue $v \leftarrow Q$
For each w such that $(v, w) \in E$
if $h_{sw} = \infty$
$h_{sw} \leftarrow h_{sw} + 1$
If $w \in Q$
Enqueue $w \rightarrow Q$
Next s.

and nm respectively represent the effort to initialize the geodesic matrix and the successive implementation of n iterations, in which at most m arcs in the network are examined.

Of note, predecessor nodes can be maintained to generate the geodesics using straightforward modifications to each of Algorithms 3, 4, and 5. The modifications necessary for such changes are included in the respective references.

3.2.4 Analysis of Measure Computations.

In the application of Social Network Analysis, many nodal and network measures can be computed using functions encoded within software tools. This analysis utilizes two suites of functions for network analysis. First, a suite of MATLAB functions was produced at MIT

and is labeled the MIT Toolbox. It was designed "for someone who wants to start hands-on work with networks fairly quickly... and compute common network theory metrics" [32]. The MIT Toolbox was hosted on MIT web servers until 2011, and it now resides online at Octave. Alternatively, the NetworkX package is a suite of functions encoded in the Python language "for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks" [38]. This package is hosted online on Github.

To demonstrate the utility of retaining and storing geodesic information and for reference, the nodal and network measures in Tables 16 and 17 are computed assuming that the geodesic information is not determined in advance. Subsequently, the geodesic information for each network considered is computed between all node pairs using both Algorithm 3 and Algorithm 4 for weighted networks and Algorithm 3 and Algorithm 5 for unweighted networks. The nodal and network measures in Tables 16 and 17 are then calculated using the retained geodesic information from each of the algorithms. The values of the measures and their respective computation times are stored for further comparison and analysis over a battery of test instances comprised of different network structures, sizes, and densities.

The calculation of geodesic-related measures is tested on three commonly utilized randomly-generated undirected network structures: Erdős-Rényi networks, Barabási-Albert networks, and Watts-Strogatz networks as described in Appendix A.

The parameters of the random networks generated to test the MIT Toolbox are listed in Table 18. For each of three network structures, 30 replicates of four test networks are generated using the parameters indicated. The density γ of a network is given by

$$\gamma = \frac{2m}{n(n-1)}.$$

Half the networks generated have density levels lower than 0.15 (low-density) and the remainder have densities larger than 0.30 (higher-density). The value of the distance weight

Table 18. Parameter settings for 100-node random network structures

Structure	Parameters	Mean Density
ER	$\beta = 0, p = 0.1$	0.098
	$\beta = 1, p = 0.1$	0.099
	$\beta = 0, p = 0.5$	0.5
	$\beta = 1, p = 0.5$	0.502
BA	$m_a = 2, n_0 = 25$	0.091
	$m_a = 5, n_0 = 25$	0.136
	$m_a = 2, n_0 = 55$	0.318
	$m_a = 10, n_0 = 50$	0.348
WS	$k = 4, p = 0.10$	0.04
	$k = 4, p = 0.25$	0.04
	$k = 30, p = 0.10$	0.303
	$k = 30, p = 0.25$	0.303

assigned to each arc in the network is selected using a discrete (integer) uniform distribution with a range between 1 and 10, inclusive.

3.2.4.1 MIT Toolbox Results.

The initial computation of nodal and network measures without retaining geodesic information is straightforward for the MIT Toolbox because the functions that compute the measures are currently encoded without utilizing retained geodesics. The MIT Toolbox measure functions are modified for this research to consider the geodesic matrix as an input. Finally, the time to compute each of the measures is recorded as well as the values of the computed nodal and network measures.

The values of the measures when computed using the standard MIT Toolbox without use of the stored geodesic matrix were compared to the modified MIT Toolbox functions that did utilize the stored geodesic matrix. Identical results were obtained on the networks tested, so the focus shifts to identifying the method that produces the results in the shortest amount of time. A summary of the time to compute the measures is provided in Table 19. The minimum, average, and maximum computation times in seconds are given when computed

using MATLAB version 2012a on an HP Compaq 6005 Pro with a 2.70 GHz AMD Athlon II X2 215 processor and 4.0 GB of RAM. The results are separated by network structure (*i.e.*, ER, BA, or WS) and density, with LD and HD indicating lower- and higher-density test networks, respectively.

The reported times are the total time required to compute each of the nodal and network measures in Tables 16 and 17. The individual measure calculation times were similar because each requires the calculation of geodesic information and uses the same computation steps to do so. In addition to the measure computation time, the time to compute the geodesic matrix is included as well. The "NR" column (no-retain method) shows the total of computation times for each of the tested measures using the unmodified MIT Toolbox without retaining the geodesic information. The next two columns provide the times for the retain methods. Column labels "RD" and "F-W" represent the tests that utilize the RD Algorithm and the F-W Algorithm to compute the geodesic matrix before using it to speed the measure calculations, respectively.

The average combined time to compute all geodesic-related measures is less than 0.01 seconds when the geodesic matrix is computed in advance. Repeatedly utilizing the the MIT Toolbox function for Dijkstra's algorithm was not as efficient as using the F-W algorithm for the networks tested. This is likely due to the simplicity of the MATLAB function evaluations in the F-W algorithm when compared with those in the MIT Toolbox implementation of the RD algorithm. Moreover, this testing indicates no tie between computation time and the network density since the average computation times fall within fractions of a second across all tests for the characteristics of the network structures tested.

In addition to testing the random networks with random distance weights as reported in Table 18, analysis of the same networks using the unweighted version of the network was conducted. The entire test for the MIT Toolbox was repeated with two changes. First, the BFS algorithm was utilized rather than the RD algorithm. Second, instead of computing the

Table 19. Average computation times for weighted, 100-node random networks using MIT Toolbox

				NR	RD	F-W
ER	LD $p = 0.1$	$\beta = 0$	min	47.485	4.961	0.042
			mean	47.960	5.107	0.043
			max	48.538	5.603	0.049
		$\beta = 1$	min	47.483	5.024	0.042
			mean	47.990	5.174	0.043
			max	48.695	5.642	0.055
	HD $p = 0.5$	$\beta = 0$	min	47.556	4.981	0.042
			mean	48.057	5.176	0.044
			max	48.694	5.603	0.061
		$\beta = 1$	min	47.518	5.015	0.042
			mean	47.992	5.174	0.043
			max	49.223	5.691	0.049
BA	LD $n_0 = 25$	$m_a = 2$	min	47.330	5.015	0.042
			mean	47.814	5.175	0.043
			max	48.501	5.549	0.047
		$m_a = \frac{n_0}{5}$	min	47.380	4.982	0.042
			mean	47.846	5.143	0.043
			max	48.622	5.667	0.048
	HD $n_0 = 55, 50$	$m_a = 2$	min	47.408	4.986	0.042
			mean	47.904	5.164	0.042
			max	49.772	5.607	0.046
		$m_a = \frac{n_0}{5}$	min	47.244	4.988	0.042
			mean	47.850	5.193	0.045
			max	48.838	5.641	0.095
WS	LD $k = 4$	$p = 0.1$	min	47.319	4.936	0.042
			mean	47.845	5.160	0.043
			max	48.443	5.465	0.055
		$p = 0.25$	min	47.302	4.995	0.042
			mean	47.908	5.233	0.042
			max	49.123	6.085	0.045
	HD $k = 30$	$p = 0.1$	min	47.425	4.946	0.042
			mean	47.930	5.134	0.043
			max	48.481	5.520	0.047
		$p = 0.25$	min	47.498	4.986	0.042
			mean	47.936	5.150	0.043
			max	48.539	5.426	0.046

Table 20. Computation times for unweighted, 100-node random networks using MIT Toolbox

				NR	BFS	F-W
ER	LD $p = 0.1$	$\beta = 0$	min	47.885	0.208	0.042
			mean	48.232	0.234	0.043
			max	48.720	0.271	0.050
		$\beta = 1$	min	48.147	0.211	0.042
			mean	49.571	0.232	0.044
			max	54.119	0.276	0.056
	HD $p = 0.5$	$\beta = 0$	min	48.106	0.682	0.042
			mean	48.396	0.742	0.045
			max	49.759	0.854	0.096
		$\beta = 1$	min	47.612	0.685	0.042
			mean	50.942	0.750	0.043
			max	59.359	0.833	0.048
BA	LD $n_0 = 25$	$m_a = 2$	min	47.633	0.198	0.041
			mean	47.983	0.218	0.043
			max	48.695	0.262	0.048
		$m_a = \frac{n_0}{5}$	min	47.930	0.251	0.041
			mean	48.516	0.275	0.043
			max	50.578	0.324	0.060
	HD $n_0 = 55, 50$	$m_a = 2$	min	47.803	0.459	0.041
			mean	48.097	0.503	0.043
			max	48.734	0.564	0.046
		$m_a = \frac{n_0}{5}$	min	47.543	0.492	0.042
			mean	47.765	0.548	0.043
			max	48.089	0.638	0.053
WS	LD $k = 4$	$p = 0.1$	min	47.509	0.144	0.042
			mean	47.886	0.159	0.044
			max	49.524	0.221	0.059
		$p = 0.25$	min	47.911	0.144	0.042
			mean	49.710	0.159	0.043
			max	56.072	0.182	0.049
	HD $k = 30$	$p = 0.1$	min	47.421	0.447	0.042
			mean	47.660	0.490	0.042
			max	48.033	0.549	0.046
		$p = 0.25$	min	48.027	0.444	0.042
			mean	52.148	0.488	0.044
			max	63.767	0.539	0.061

distance measures using the weighted value of distance, the unweighted adjacency matrix was selected, *i.e.*, $D = A$. This modification alters the interpretation of the geodesic matrix. Matrix entries instead represent the number of links between nodes. The computation time results are summarized in Table 20.

The average combined time to compute all geodesic-related measures is 0.01 seconds. As expected, the modification of the distance weights to values of 1 did not change the result that measure values were equivalent for all tests. The MIT Toolbox computation times are examined to identify which method is preferred. The computation times corresponding to the BFS algorithm indicate that algorithm performance depends on the density of the matrix. Each of the higher-density network instances tested had larger computation times as evidenced by the results reported in the HD rows of Table 20. The implementation of the F-W algorithm is slightly faster than the BFS algorithm when executed in MATLAB in these tests. The testing of the Python-based NetworkX package will further assess this observation.

For the MIT Toolbox, significant improvement in the computation of geodesic related measures can be realized by modifying the available MATLAB code. Improvements will be realized by implementing two changes. First, implement a routine to preprocess and store the geodesic matrix. Second, modify the geodesic related measures to perform the appropriate matrix manipulation to compute the measure based on the geodesic matrix.

3.2.4.2 NetworkX Package Results.

The geodesic-related measures are also computed using the NetworkX package. As with the MIT Toolbox, the values of the measures when considering both with and without retaining information are equivalent for the networks tested, so the analysis focuses on computation time. The minimum, average, and maximum computation times using Python version 2.7 and NetworkX version 1.9.1 on the same computational platform are listed in

Table 21. Again, the majority of the time is used to compute the geodesic matrix for the retain methods. The average combined time to compute all geodesic related measures is 0.03 seconds. Using the RD algorithm already embedded within the NetworkX package was more efficient than the F-W algorithm on the lower density networks tested. This confirms the conventional wisdom that the F-W algorithm should be used for dense networks. The networks that have a density larger than 0.35 (*i.e.*, ER with $p = 0.5$) appear to have faster F-W times. In fact, for the networks tested, the F-W times in MATLAB were faster than those in Python. This may be attributed to the fact that the operation is so simple for each iteration in the algorithm that MATLAB is able to solve it very quickly.

The random networks in Table 18 were also tested using their unweighted versions. The entire test for the NetworkX package was repeated utilizing the unweighted adjacency matrix rather than the distance matrix, *i.e.*, $D = A$. The selection of the BFS algorithm is handled within the NetworkX code when computing the geodesic matrix for unweighted networks. The minimum, average, and maximum computation time results for the tested networks are summarized in Table 22. The average combined time to compute all geodesic related measures is 0.03 seconds. Because of the internal handling of the faster BFS algorithm in computing geodesics, the average computation times without retention were faster than when considering the weighted networks. It is again apparent that the no-retain and BFS algorithms are dependent on the density of the network. Meanwhile the F-W computation times remained invariant relative to network density and/or network structure.

Figure 8 depicts the measure calculation times and algorithm completion times for all 100-node network instances tested. For the networks tested, there appears to be a density threshold at which the F-W algorithm becomes more efficient near density level 0.4. The figure also shows the rapid measure calculation times for NetworkX. The calculation times that appear to be outliers are likely due to a network application running at the same time

Table 21. Computation times for weighted, 100-node random networks using NetworkX

				NR	RD	F-W
ER	LD $p = 0.1$	$\beta = 0$	min	1.353	0.125	0.386
			mean	1.643	0.146	0.445
			max	2.088	0.223	0.609
		$\beta = 1$	min	1.271	0.124	0.395
			mean	1.439	0.136	0.444
			max	1.763	0.175	0.587
	HD $p = 0.5$	$\beta = 0$	min	5.185	0.484	0.392
			mean	5.891	0.554	0.441
			max	7.633	0.734	0.614
		$\beta = 1$	min	5.172	0.486	0.396
			mean	5.465	0.552	0.460
			max	5.967	0.689	0.651
BA	LD $n_0 = 25$	$m_a = 2$	min	1.212	0.116	0.378
			mean	1.547	0.127	0.421
			max	1.908	0.250	0.538
		$m_a = \frac{n_0}{5}$	min	1.797	0.157	0.385
			mean	1.952	0.175	0.437
			max	2.220	0.348	0.604
	HD $n_0 = 55, 50$	$m_a = 2$	min	3.286	0.303	0.379
			mean	3.978	0.346	0.429
			max	4.879	0.514	0.602
		$m_a = \frac{n_0}{5}$	min	3.471	0.328	0.365
			mean	4.227	0.368	0.422
			max	5.160	0.558	0.580
WS	LD $k = 4$	$p = 0.1$	min	0.688	0.070	0.381
			mean	0.759	0.075	0.433
			max	0.886	0.090	0.594
		$p = 0.25$	min	0.709	0.073	0.385
			mean	0.855	0.077	0.440
			max	1.544	0.089	0.600
	HD $k = 30$	$p = 0.1$	min	3.246	0.305	0.388
			mean	3.465	0.341	0.454
			max	4.131	0.453	0.635
		$p = 0.25$	min	3.177	0.313	0.394
			mean	3.485	0.348	0.457
			max	4.575	0.506	0.696

Table 22. Computation times for unweighted, 100-node random networks using NetworkX

				NR	BFS	F-W
ER	LD $p = 0.1$	$\beta = 0$	min	0.189	0.125	0.398
			mean	0.205	0.138	0.477
			max	0.272	0.178	0.700
		$\beta = 1$	min	0.195	0.127	0.394
			mean	0.249	0.139	0.445
			max	0.351	0.182	0.559
	HD $p = 0.5$	$\beta = 0$	min	0.343	0.490	0.397
			mean	0.402	0.549	0.456
			max	0.481	0.648	0.678
		$\beta = 1$	min	0.342	0.489	0.393
			mean	0.423	0.552	0.449
			max	0.564	0.691	0.646
BA	LD $n_0 = 25$	$m_a = 2$	min	0.168	0.116	0.381
			mean	0.188	0.124	0.447
			max	0.253	0.150	0.552
		$m_a = \frac{n_0}{5}$	min	0.192	0.157	0.392
			mean	0.208	0.176	0.466
			max	0.240	0.238	0.705
	HD $n_0 = 55, 50$	$m_a = 2$	min	0.220	0.303	0.389
			mean	0.245	0.338	0.459
			max	0.341	0.425	0.641
		$m_a = \frac{n_0}{5}$	min	0.245	0.330	0.371
			mean	0.273	0.363	0.440
			max	0.349	0.442	0.604
WS	LD $k = 4$	$p = 0.1$	min	0.170	0.071	0.385
			mean	0.182	0.077	0.434
			max	0.219	0.098	0.616
		$p = 0.25$	min	0.169	0.072	0.394
			mean	0.183	0.076	0.441
			max	0.249	0.103	0.546
	HD $k = 30$	$p = 0.1$	min	0.252	0.307	0.394
			mean	0.301	0.347	0.443
			max	0.381	0.443	0.519
		$p = 0.25$	min	0.259	0.315	0.395
			mean	0.303	0.356	0.449
			max	0.421	0.461	0.553

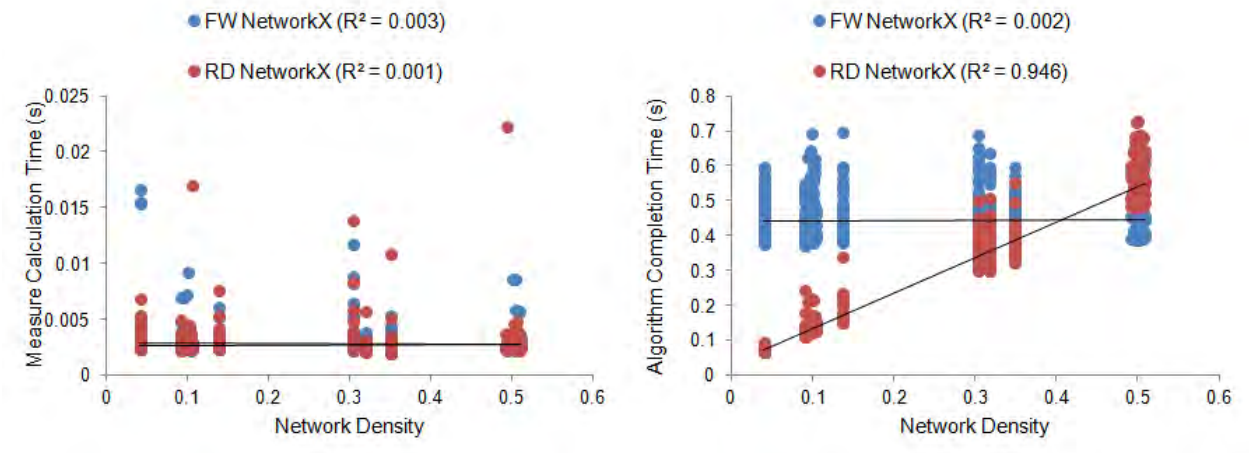


Figure 8. Measure calculation times and algorithm completion times for all 100-node test instances

and reducing the processing power during the calculation being completed. There is no evidence that a specific network topology caused the longer calculation times.

Interestingly, the MATLAB F-W implementation for calculating the (weighted and unweighted) geodesic matrix required less time than any of the NetworkX computations for the network instances tested. The speed with which MATLAB can complete all iterations combined with the relatively simple evaluations in the F-W algorithm make it a good choice when calculating a geodesic matrix in advance.

3.2.4.3 Additional Testing Results.

The NetworkX implementation of the F-W and RD Algorithms and the MATLAB F-W implementation are further tested using 1,000-node networks that are constructed according to the parameters listed in Table 23. Because the networks are larger and the computation time will increase, 10 replicates of each of the 1,000-node networks are studied.

For the weighted 1,000-node tests, the minimum, average, and maximum computation times are listed in Table 24. The combined computation time for all geodesic related measures after the geodesic matrix is computed was 0.3 seconds in MATLAB and 0.15 seconds

in NetworkX. The F-W algorithm implemented in MATLAB (*i.e.*, column "F-W (M)") consistently was measured at just over one minute for each network structure. The initialization in NetworkX iterates through each arc to update the distances. MATLAB uses the distance matrix to initialize the values. This difference accounts for the slight variation in F-W results for NetworkX that are not present in the MATLAB results. The smallest densities tested (*i.e.*, $\gamma \approx 0.04$ for the low-density WS network structure and the low-density BA network structure for $m_a = 2$) appear to be near the threshold at which the NetworkX code is faster than the MATLAB code for computing the geodesic matrix for the tested network instances.

Similar conclusions arise when testing unweighted 1,000-node networks, for which results are summarized in Table 25. The average total computation time for the MATLAB implementation of the geodesic measures is 0.03 seconds. The NetworkX average total computation time for the measures is 0.16 seconds.

The average computation times are larger than the 100-node tests; this extended length of time required to compute the geodesic matrix necessitates its processing in advance. In all combinations of weighted or unweighted 1,000-node network test cases and analysis using MATLAB or NetworkX, it is clear that processing the geodesic matrix in advance enables

Table 23. Parameter settings for 1,000-node random network structures

Structure	Parameters	mean Density
ER	$\beta = 0, p = 0.1$	0.1
	$\beta = 1, p = 0.1$	0.1
	$\beta = 0, p = 0.5$	0.5
	$\beta = 1, p = 0.5$	0.5
BA	$m_a = 2, n_0 = 250$	0.065
	$m_a = 50, n_0 = 250$	0.137
	$m_a = 2, n_0 = 550$	0.304
	$m_a = 100, n_0 = 500$	0.35
WS	$k = 40, p = 0.10$	0.04
	$k = 40, p = 0.25$	0.04
	$k = 300, p = 0.10$	0.303
	$k = 300, p = 0.25$	0.303

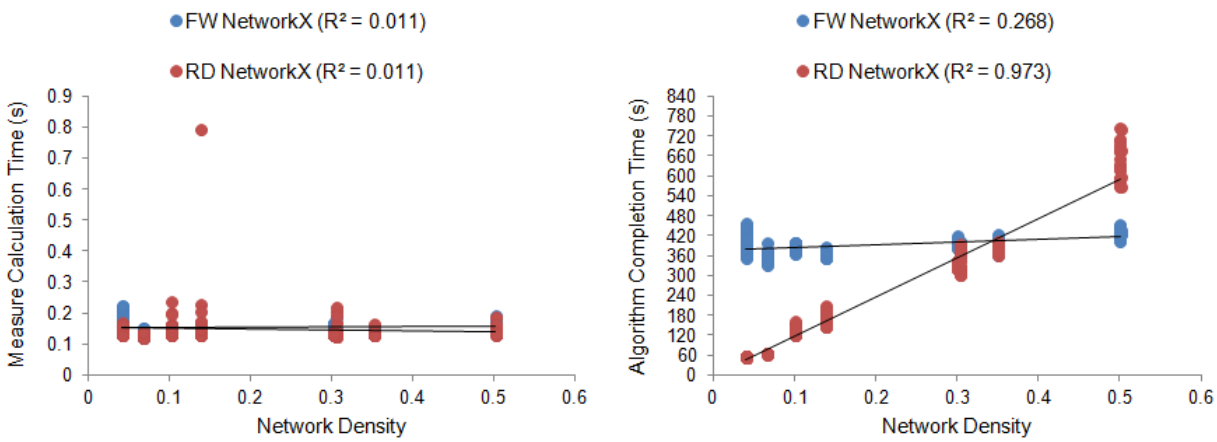


Figure 9. Measure calculation times and algorithm completion times for all 1,000-node test instances

rapid measure computations. The unweighted computation times are very similar to their weighted counterparts, indicating that the number of nodes and the density affect how quickly the geodesic matrix is computed.

Figure 9 depicts the measure calculation times and algorithm completion times for all 1,000-node network instances tested. As with the 100-node test, there appears to be a density threshold at which the F-W algorithm becomes more efficient near density level 0.35, which is less than the density level for the smaller network instances tested. The figure also shows the rapid measure calculation times for NetworkX. The calculation time that appears to be an outlier is likely due to a network application running at the same time and reducing the processing power during the calculation being completed. There is no evidence that a specific network topology caused the longer calculation time.

3.2.5 Geodesics and Related Measures Summary.

This section addresses the importance of the effectiveness that may be achieved by preprocessing the geodesic matrix. Without understanding the relationship between the geodesic matrix and the measures related to geodesic lengths, the computation time for larger net-

Table 24. Computation times for weighted, 1,000-node random networks using NetworkX and MATLAB

				RD	F-W	F-W (M)
ER	LD $p = 0.1$	$\beta = 0$	min	121.146	382.320	65.991
			mean	126.184	388.703	66.060
			max	130.451	393.977	66.271
		$\beta = 1$	min	123.210	385.858	65.973
			mean	129.691	396.762	66.057
			max	133.053	403.566	66.378
	HD $p = 0.5$	$\beta = 0$	min	566.333	418.471	65.867
			mean	581.140	426.256	65.935
			max	598.804	435.398	66.141
		$\beta = 1$	min	570.912	420.922	65.818
			mean	582.497	427.996	65.946
			max	591.486	437.405	66.267
BA	LD $n_0 = 250$	$m_a = 2$	min	64.173	349.491	65.741
			mean	65.960	358.467	65.866
			max	67.562	378.050	66.058
		$m_a = \frac{n_0}{5}$	min	144.951	364.881	65.887
			mean	149.589	370.526	66.016
			max	154.824	374.493	66.172
	HD $n_0 = 550, 500$	$m_a = 2$	min	304.094	373.580	65.751
			mean	313.467	383.453	65.842
			max	327.125	389.820	66.041
		$m_a = \frac{n_0}{5}$	min	365.724	371.470	65.853
			mean	383.806	387.133	66.425
			max	404.628	397.098	70.620
WS	LD $k = 40$	$p = 0.1$	min	52.674	367.339	65.846
			mean	53.782	399.021	65.935
			max	54.815	444.351	66.132
		$p = 0.25$	min	54.288	384.497	65.886
			mean	55.745	411.045	65.970
			max	56.544	454.563	66.237
	HD $k = 300$	$p = 0.1$	min	317.594	396.147	65.871
			mean	331.351	405.417	66.114
			max	344.783	424.043	66.499
		$p = 0.25$	min	324.773	397.818	65.878
			mean	336.851	406.781	66.019
			max	344.34	415.931	66.315

Table 25. Computation times for unweighted, 1,000-node random networks using NetworkX and MATLAB

				RD	F-W	F-W (M)
ER	LD $p = 0.1$	$\beta = 0$	min	136.214	370.352	66.418
			mean	144.521	375.051	66.535
			max	156.744	386.072	66.845
		$\beta = 1$	min	128.469	382.724	66.336
			mean	139.742	389.967	66.518
			max	161.718	401.843	66.800
	HD $p = 0.5$	$\beta = 0$	min	636.096	407.352	66.371
			mean	695.516	434.486	66.548
			max	744.977	455.426	66.737
		$\beta = 1$	min	580.476	409.224	66.331
			mean	602.993	427.026	66.513
			max	651.726	453.788	66.837
BA	LD $n_0 = 250$	$m_a = 2$	min	63.863	335.996	66.296
			mean	65.617	360.781	66.485
			max	67.137	400.868	66.691
		$m_a = \frac{n_0}{5}$	min	166.060	356.603	66.311
			mean	187.276	369.737	66.549
			max	211.089	388.562	66.982
	HD $n_0 = 550, 500$	$m_a = 2$	min	303.462	360.831	66.245
			mean	348.171	385.154	66.485
			max	395.350	406.822	66.867
		$m_a = \frac{n_0}{5}$	min	359.996	366.237	66.312
			mean	373.322	391.721	66.905
			max	385.173	428.940	68.751
WS	LD $k = 40$	$p = 0.1$	min	52.924	355.507	66.369
			mean	54.061	386.223	66.538
			max	54.775	430.908	66.775
		$p = 0.25$	min	54.242	386.502	66.316
			mean	55.906	413.888	66.517
			max	56.757	461.286	67.062
	HD $k = 300$	$p = 0.1$	min	319.127	386.253	66.325
			mean	325.734	396.380	66.477
			max	335.543	414.320	66.717
		$p = 0.25$	min	327.608	387.298	66.382
			mean	336.479	396.909	66.678
			max	343.287	405.209	67.435

works may become prohibitive for practical purposes. In contrast, a proper application of algorithms that provide such geodesic information allows an analyst to more quickly assess the impact of geodesic lengths to characterize the importance of nodes within the network using nodal and/or network measures. A knowledge of the relationship between the geodesic matrix and measure computations allows the implementation of much more efficient code. Moreover, knowledge of the detailed implementation of an algorithm to compute measures allows for enhancements to the efficiency with which it is encoded, thereby reducing the time required to attain key results.

Future Work. As indicated in the analysis of the results of the NetworkX testing, there appears to be a density threshold at which the F-W algorithm becomes more efficient (for these tests approximately when density is between 0.3 and 0.4). However, in the open literature, this threshold has not been explored. "There is no strict distinction between sparse and dense [networks]" [8]. General guidance indicates that the F-W algorithm is appropriate for dense graphs and RD-type algorithms are more suited to sparse graphs. A more rigorously designed experiment and analysis of the NetworkX code for computing the geodesic matrix using the two methods over appropriately selected random networks with varying densities would provide a more accurate value of such a threshold.

3.3 Extending All Geodesics Information

In the previous section, the utility of computing lengths of geodesics, or shortest paths, for all node pairs in a network was demonstrated. Because preprocessing all geodesic lengths enables nodal and network measures to be computed rapidly, the algorithms that compute geodesic lengths for all node pairs are extended in this section to include more output information so additional measures may be rapidly calculated.

This section proposes and demonstrates the utility of storing additional information about network features to enable the rapid calculation of several nodal and network measures after the information is preprocessed. The algorithms that generated the geodesic matrix to reduce computation time are leveraged when constructing the Extended All Geodesic Lengths Algorithm (EAGL) to account for more features. The betweenness centrality measure is used to demonstrate its usefulness. Analysis of measure computations on randomly generated networks, both with and without the stored information, are compared and contrasted to demonstrate the importance of preprocessing and retaining the geodesic and dependency information for all node pairs.

3.3.1 Extending Geodesic Algorithms.

The algorithms presented in the previous section collect geodesic information to calculate several nodal and network measures. These measures are computed rapidly using the geodesic matrix containing the lengths of the geodesics between all node pairs. Unfortunately, there are a number of measures for which the computation requires more information than is contained in the geodesic matrix. One such measure is betweenness.

The *betweenness centrality* of a node $C_B(i)$ measures the extent to which nodes are along the geodesic between node pairs. These nodes that are between others and along their geodesic may have more influence over the others and they have the potential to affect the flow of information along the path. The betweenness centrality of node k is [10, 53, 63]

$$C_B(k) = \sum_{i \neq k \neq j \in V} \frac{n_{ij}(k)}{n_{ij}},$$

where n_{ij} is the number of (i, j)-geodesics, and $n_{ij}(k)$ is the number of (i, j)-geodesics that include node k. The betweenness centrality can be computed whether the network is connected

and/or directed [63]. The betweenness centrality values can also be computed whether the arc distances have a length of 1 or some other value [10].

The betweenness centrality for each node in a network is readily computed using the Betweenness Centrality Algorithm given by Brandes [10]. The algorithm runs in $O(nm)$ time for networks with unweighted arcs and $O(nm + n^2 \log n)$ time for weighted networks using a Fibonacci heap. Algorithm 6 details the unweighted version of the procedure to compute betweenness centralities. The algorithm utilizes a geodesic discovery and counting routine (BFS) for each node and then accumulates the betweenness centrality using node dependencies. Node dependencies are computed using a recursive relation (applied in the accumulation phase of the algorithm) so that n_{ij} and $n_{ij}(k)$ in the betweenness calculation are not directly computed. The algorithm presented is as set forth by Brandes [11].

Algorithm 6 is extended to store the geodesic information for use in subsequent nodal and network measure calculations. The Extended All Geodesics Lengths (EAGL) Algorithm, which is stated in Algorithm 7, takes the features of the aforementioned algorithms (*i.e.*, retaining all geodesic and predecessor information from the geodesic algorithms (*e.g.*, the F-W and RD Algorithms) as well as computing node dependencies and counting occurrences of nodes on geodesics from Algorithm 6) to construct a more robust algorithm. Utilizing the output geodesic lengths and node dependencies, the nodal and network measures related to these features can be quickly computed. The algorithm can be further extended to accommodate the accumulation of other features of the network to enable rapid calculation of additional measures.

To expand the Betweenness Centrality Algorithm of Brandes [10, 11], the arrays are replaced by matrices to maintain the geodesic lengths between each node pair, the number of geodesics on which each node appears, and all predecessors nodes for node pairs. The associated increase in storage space requirements for the algorithm will affect is speed when

Algorithm 6 Betweenness Centrality Algorithm (unweighted) [11:p. 5]

Input

A directed network with nodes $V = \{1, \ldots, n\}$ and arcs $E = (i, j), i, j \in V$.

Data

Q: queue for nodes.
S: stack for nodes.
dist(v): distance from source.
Pred(w): list of predecessors on geodesics from source.
$\sigma(v)$: number of geodesics from source to node v.
$\delta(v)$: dependency of source on node v.

Output

The betweenness centrality $C_B(i)$ for each node i.

Algorithm

For $s \in V$,
  Single-Source Geodesics Problem
  Initialization
    For $t \in V$,
      Pred(w) $\leftarrow$ empty list; dist(t) $\leftarrow \infty$; $\sigma(t) \leftarrow 0$
    dist(s) $\leftarrow 0$; $\sigma(s) \leftarrow 1$
    Enqueue $s \rightarrow Q$
  While Q not empty
    Dequeue $v \leftarrow Q$; Push $v \rightarrow S$
    For each w such that $(v, w) \in E$
      Path Discovery
      if dist(w) $= \infty$
        dist(w) $\leftarrow$ dist(v) $+ 1$; Enqueue $w \rightarrow Q$
      Path Counting
      if dist(w) $=$ dist(v) $+ 1$
        $\sigma(w) \leftarrow \sigma(w) + \sigma(v)$; Append $v \rightarrow$ Pred(w)
  Accumulation
    For $v \in V$, $\delta(v) \leftarrow 0$
    While S not empty
      Pop $w \leftarrow S$
      For $v \in$ Pred(w)
        $\delta(v) \leftarrow \delta(v) + \frac{\sigma(v)}{\sigma(w)}(1 + \delta(w))$
      If $w \neq s$, $C_B(w) \leftarrow C_B(w) + \delta(w)$
Next s.

Algorithm 7 Extended All Geodesics Lengths (EAGL) Algorithm

Input

Directed network with nodes $V = \{1, \ldots, n\}$ and arcs $E = (i, j), i, j \in V$.
A: matrix of adjacency between node pairs.
D: matrix of distances between node pairs.

Data

Q: FIFO queue if unweighted, priority queue if weighted.
S: stack for nodes.

Output

G: matrix of geodesics between node pairs.
M: matrix of dependencies between node pairs.
N: matrix of number of geodesics between node pairs.
P: matrix of lists of predecessors on geodesics between node pairs.

Algorithm

Initialization
For $s, t \in V$,
$g_{st} \leftarrow \infty$; $m_{st} \leftarrow 0$; $n_{st} \leftarrow 0$; $p_{st} \leftarrow$ empty list; $\delta_{st} \leftarrow 0$; $\sigma_{st} \leftarrow 0$;
Single-Source Geodesics Problem
For $s \in V$,
$g_{ss} \leftarrow 0$; $\sigma_{ss} \leftarrow 1$
If unweighted
BFS
Else
Dijkstra
Accumulation
While S not empty
Pop $w \leftarrow S$
For $v \in p_{sw}$
$m_{sv} \leftarrow m_{sv} + \frac{n_{sv}}{n_{sw}}(1 + m_{sw})$
Next s.

data points are accessed from very large matrices. The increased storage requirements do not, however, alter the required number of operations, other than placing data into storage.

The general outline of the EAGL Algorithm (Algorithm 7) follows that of the Betweenness Centrality Algorithm (Algorithm 6). First geodesics from node i to all other nodes and their counts are determined using BFS for unweighted networks and Dijkstra's Algorithm for weighted networks. Then, the dependency of node i on every other node is accumulated

BFS

Enqueue $s \rightarrow Q$
While Q not empty
  Dequeue $v \leftarrow Q$; Push $v \rightarrow S$
  For each w such that $(v, w) \in E$
    Path Discovery
    if $g_{sw} = \infty$
      $g_{sw} \leftarrow g + 1$; Enqueue $w \rightarrow Q$
    Path Counting
    if $g_{sw} = g_{sv} + 1$
      $n_{sw} \leftarrow n_{sw} + n_{sv}$; Append $v \rightarrow p_{sw}$

Dijkstra

Enqueue $s \rightarrow Q$ with priority 0
While Q not empty
  Dequeue minimum priority $v \leftarrow Q$ with $d \leftarrow$ priority
  Push $v \rightarrow S$
  For each w such that $(v, w) \in E$
    Path Discovery
    dist $\leftarrow d + d_{vw}$
    if $g_{sw} >$ dist
      $g_{sw} \leftarrow$ dist; $n_{sw} \leftarrow n_{sv}$; $p_{sw} \leftarrow v$
      If $w \in Q$
        Update $w \in Q$ to priority dist
      Else
        Enqueue $w \rightarrow Q$ with priority dist
    Path Counting
    Else if $g_{sw} =$ dist
      $n_{sw} \leftarrow n_{sw} + n_{sv}$; Append $v \rightarrow p_{sw}$

---

with the order reversed from the order of the discovery of its geodesic from i. The EAGL Algorithm retains the dependencies in matrix form, allowing the betweenness centrality to be computed upon termination.

The EAGL Algorithm outputs several matrices that are used to compute the measures. The geodesic matrix G has entries g_{ij} representing the length of the geodesic from node i to node j. The number of geodesics between node i and node j is represented by entry (i, j) in the geodesic count matrix N. For each node pair, the list of immediate predecessors is contained in the matrix P, where p_{ij} is the list of immediate predecessors of node j on the

(i, j)-geodesic. The dependency matrix M, where the value m_{ij} indicates the dependency of node i on node j, contains the dependency between node pairs required for the betweenness centrality calculation. Finally, Algorithm 6 is further modified by removing the betweenness centrality calculation.

The betweenness centrality is computed using the matrix M directly, after the algorithm has run to completion. The calculation of the betweenness centrality of node i using the M matrix output of the EAGL Algorithm is given as

$$C_B(j) = \sum_{i \neq j \in V} m_{ij}. \tag{3.1}$$

In other words, the betweenness centrality measure is computed by storing the dependencies and then adding them after all iterations rather than adding them during each iteration of the algorithm.

3.3.2 Analysis of Measure Computations.

To demonstrate the utility of retaining geodesics and dependency information, several nodal measures are computed without implementing the EAGL Algorithm in advance. The measures are calculated within the NetworkX package for the Python coding language. The nodal measures utilized in this test are listed in Table 26. Subsequently, network measures are computed using the appropriate nodal measure values as inputs. The network measures and corresponding nodal measure used as their respective input are denoted in Table 27. Computation times for each calculation are recoded for analysis.

The test networks are then evaluated using the EAGL Algorithm as a preprocessing step. The time to complete the EAGL Algorithm is recorded, then the nodal measures are calculated using as input both the geodesic and dependency outputs of the algorithm. The

network measures are then computed in the same manner as before, the computation times are recorded for analysis.

The test employs the same test cases utilized and described in the previous section. Recall that for each of three network structures, 30 replicates of four test networks are generated using the parameters indicated for the 100-node instances and 10 replicates are utilized for the 1,000-node instances. Half of the test networks have density levels, γ, less than 0.15 (low-density), and the remainder have densities larger than 0.30 (higher-density). The parameters of the random networks used in testing the EAGL Algorithm implementations are again listed in Tables 18 and 23. The value of the distance-weight assigned to each arc in the network is selected using a discreet (integer) uniform distribution having a range between 1 and 10, inclusively.

Table 26. Nodal measures and inputs

Nodal Measure	Input from EAGL
eccentricity	geodesic
total distance	geodesic
closeness centrality	geodesic
betweenness centrality	dependency

Table 27. Network measures and inputs

Network Measure	Input Nodal Measure
diameter	eccentricity
radius	eccentricity
central nodes	eccentricity
peripheral nodes	eccentricity
medial nodes	total distance
Wiener index	total distance
average distance	total distance

3.3.2.1 100-Node Testing Results.

The baseline computation of nodal and network measures without preprocessing using the EAGL Algorithm is completed using NetworkX. Moreover, the comparison computations of nodal and network measures using the outputs of the EAGL Algorithm are completed using NetworkX. Finally, the EAGL Algorithm is implemented in MATLAB for comparison. The values of the measures when computed are equivalent, so the results discussed focus on computation times. A summary of the times to compute the measures for weighted networks is given in Table 28 and the unweighted network test results are listed in Table 29. The minimum, average, and maximum computation times in seconds are given when computed using Python version 2.7, NetworkX version 1.9.1, and MATLAB version 2012a on an HP Compaq 6005 Pro having a 2.70 GHz AMD Athlon II X2 215 processor and 4.0 GB of RAM. The results are separated by network type (*i.e.*, ER, BA, or WS) and density, with lower- and higher-density test networks indicated by LD and HD, respectively. The NetworkX calculations without using the EAGL Algorithm are listed in the "NetworkX" column of the table. The computation times using the EAGL Algorithm as a preprocessing step via the NetworkX package and MATLAB are reported in columns "EAGL (NetX)" and "EAGL (MATLAB)", respectively.

Comparisons are considered between the NetworkX tests (*i.e.*, columns "NetworkX" and "EAGL (NetworkX)") to draw inferences about the suitability of the EAGL Algorithm to improve computation times. For the weighted, 100-node networks, the implementation of the EAGL Algorithm to preprocess geodesic and dependency information resulted in computation times roughly half those without preprocessing for lower-density networks tested and nearly a third of the computation time for the tested higher-density network instances. Interestingly, for the unweighted, 100-node networks, the baseline NetworkX code was faster than the NetworkX implementation after computing the EAGL Algorithm in advance for the networks tested. This is likely because the nodal measures each use the BFS Algorithm in-

Table 28. Computation times for weighted, 100-node random networks for EAGL Algorithm testing

				NetworkX	EAGL (NetworkX)	EAGL (MATLAB)
ER	LD $p = 0.1$	$\beta = 0$	min	0.492	0.236	0.951
			mean	0.520	0.247	0.995
			max	0.545	0.276	1.081
		$\beta = 1$	min	0.493	0.237	0.946
			mean	0.518	0.250	0.976
			max	0.540	0.293	1.117
	HD $p = 0.5$	$\beta = 0$	min	1.883	0.623	2.566
			mean	1.919	0.640	2.621
			max	1.958	0.708	2.691
		$\beta = 1$	min	1.886	0.590	2.553
			mean	1.935	0.634	2.614
			max	2.002	0.750	2.723
BA	LD $n_0 = 25$	$m_a = 2$	min	0.472	0.223	0.891
			mean	0.489	0.234	0.906
			max	0.520	0.253	0.918
		$m_a = \frac{n_0}{5}$	min	0.627	0.271	1.125
			mean	0.643	0.283	1.163
			max	0.662	0.321	1.309
	HD $n_0 = 55, 50$	$m_a = 2$	min	1.190	0.437	1.804
			mean	1.212	0.456	1.845
			max	1.239	0.511	1.929
		$m_a = \frac{n_0}{5}$	min	1.333	0.449	1.956
			mean	1.361	0.470	2.007
			max	1.403	0.501	2.130
WS	LD $k = 4$	$p = 0.1$	min	0.285	0.171	0.663
			mean	0.293	0.178	0.676
			max	0.303	0.202	0.699
		$p = 0.25$	min	0.293	0.171	0.661
			mean	0.300	0.181	0.676
			max	0.310	0.204	0.710
	HD $k = 30$	$p = 0.1$	min	1.229	0.422	1.794
			mean	1.259	0.438	1.821
			max	1.320	0.457	1.850
		$p = 0.25$	min	1.232	0.436	1.794
			mean	1.253	0.454	1.828
			max	1.283	0.489	1.868

Table 29. Computation times for unweighted, 100-node random networks for EAGL Algorithm testing

				NetworkX	EAGL (NetworkX)	EAGL (MATLAB)
ER	LD $p = 0.1$	$\beta = 0$	min	0.121	0.180	0.940
			mean	0.125	0.192	0.976
			max	0.129	0.223	1.010
		$\beta = 1$	min	0.121	0.182	0.907
			mean	0.125	0.207	0.957
			max	0.130	0.250	1.041
	HD $p = 0.5$	$\beta = 0$	min	0.304	0.621	3.129
			mean	0.316	0.653	3.179
			max	0.346	0.713	3.244
		$\beta = 1$	min	0.298	0.663	3.097
			mean	0.311	0.757	3.171
			max	0.329	0.857	3.447
BA	LD $n_0 = 25$	$m_a = 2$	min	0.106	0.163	0.826
			mean	0.109	0.176	0.841
			max	0.120	0.235	0.868
		$m_a = \frac{n_0}{5}$	min	0.130	0.217	1.094
			mean	0.135	0.227	1.116
			max	0.160	0.254	1.141
	HD $n_0 = 55, 50$	$m_a = 2$	min	0.159	0.345	1.496
			mean	0.167	0.365	1.524
			max	0.191	0.541	1.552
		$m_a = \frac{n_0}{5}$	min	0.189	0.417	1.912
			mean	0.197	0.454	2.024
			max	0.216	0.549	2.309
WS	LD $k = 4$	$p = 0.1$	min	0.097	0.123	0.664
			mean	0.099	0.128	0.727
			max	0.103	0.148	0.857
		$p = 0.25$	min	0.098	0.123	0.647
			mean	0.102	0.129	0.674
			max	0.110	0.146	0.830
	HD $k = 30$	$p = 0.1$	min	0.197	0.395	1.775
			mean	0.206	0.448	1.961
			max	0.218	0.521	2.479
		$p = 0.25$	min	0.198	0.403	1.857
			mean	0.204	0.414	1.950
			max	0.215	0.435	2.318

ternally in the NetworkX code, whereas the EAGL Algorithm also utilizes BFS but requires additional steps for accumulation, storing, and retrieving the geodesic information. This may make the EAGL Algorithm for unweighted networks less efficient than when considered with arc weights. However, deliberate planning decision makers and analysts may benefit from the ability to preprocess the EAGL Algorithm to attain the relevant information and compute measure values in fractions of a second.

When considering either the weighted or unweighted 100-node tests, the total time to compute the measures was respectively less than 0.006 seconds and 0.016 seconds when preprocessing the EAGL Algorithm in Python and MATLAB. The relationship between the Python (NetworkX) and MATLAB implementation of the EAGL Algorithm is more consistent for weighted and unweighted network instances. The MATLAB implementation is consistently slower than the Python implementation by seconds for higher-density networks. The difference is not as large for lower-density networks, but the Python implementation is still faster for the tested networks. Thus, network density impacts the speed with which the EAGL Algorithm computes the geodesic and dependency information in both Python and MATLAB.

3.3.2.2 1,000-Node Testing Results.

The baseline computation times of nodal and network measures without implementing the EAGL Algorithm in advance and the computation times utilizing the EAGL Algorithm outputs are further tested using 1,000-node networks. Because the networks are larger and the computation time will increase, 10 replicates of the 1,000-node networks are studied.

For the weighted, 1,000-node test instances, the minimum, average, and maximum computation times are listed in Table 30. The total computation time for the nodal and network measures without executing the EAGL Algorithm was as given for the total computation time in the "NetworkX" column within Table 30. In the test, the combined computation time

Table 30. Computation times for weighted, 1,000-node random networks for EAGL Algorithm testing

				NetworkX	EAGL (NetworkX)	EAGL (MATLAB)
ER	LD $p = 0.1$	$\beta = 0$	min	545.097	188.552	532.573
			mean	568.067	196.426	535.023
			max	580.719	201.342	540.122
		$\beta = 1$	min	540.498	197.904	532.208
			mean	564.697	207.875	535.177
			max	579.561	214.640	543.396
	HD $p = 0.5$	$\beta = 0$	min	2428.917	669.203	2113.944
			mean	2482.451	695.608	2128.024
			max	2534.796	710.374	2154.880
		$\beta = 1$	min	2461.556	673.915	2374.696
			mean	2548.988	699.740	2579.201
			max	2637.119	717.622	2764.166
BA	LD $n_0 = 250$	$m_a = 2$	min	282.389	99.648	355.571
			mean	286.292	107.206	358.648
			max	291.501	118.446	366.387
		$m_a = \frac{n_0}{5}$	min	642.869	202.848	676.921
			mean	665.001	212.967	679.576
			max	682.312	221.237	685.432
	HD $n_0 = 550, 500$	$m_a = 2$	min	1330.465	358.121	1279.623
			mean	1373.583	362.864	1371.711
			max	1434.382	369.856	1514.449
		$m_a = \frac{n_0}{5}$	min	1611.936	454.405	1522.401
			mean	1700.565	459.331	1529.309
			max	1939.163	463.953	1535.249
WS	LD $k = 40$	$p = 0.1$	min	234.326	93.644	273.551
			mean	245.775	97.895	275.004
			max	258.201	102.991	278.711
		$p = 0.25$	min	234.750	96.751	277.141
			mean	246.849	102.489	278.946
			max	255.372	106.420	281.971
	HD $k = 300$	$p = 0.1$	min	1344.966	385.196	1324.347
			mean	1402.696	398.637	1433.827
			max	1484.916	409.461	1673.532
		$p = 0.25$	min	1452.170	409.919	1343.801
			mean	1485.022	440.260	1354.931
			max	1534.871	514.950	1394.945

Table 31. Computation times for unweighted, 1,000-node random networks for EAGL Algorithm testing

				NetworkX	EAGL (NetworkX)	EAGL (MATLAB)
ER	LD $p = 0.1$	$\beta = 0$	min	100.417	134.961	481.684
			mean	108.156	138.479	484.009
			max	112.844	140.351	487.264
		$\beta = 1$	min	103.052	138.875	485.629
			mean	110.128	144.300	503.609
			max	112.966	161.031	647.436
	HD $p = 0.5$	$\beta = 0$	min	523.189	694.632	2990.897
			mean	538.894	713.896	3226.610
			max	556.960	724.064	3365.102
		$\beta = 1$	min	509.540	699.929	3060.777
			mean	531.826	723.613	3334.784
			max	541.691	733.572	3907.739
BA	LD $n_0 = 250$	$m_a = 2$	min	39.865	73.637	292.595
			mean	40.646	75.563	295.424
			max	41.778	84.336	297.018
		$m_a = \frac{n_0}{5}$	min	109.919	168.967	679.224
			mean	115.210	173.337	680.555
			max	121.263	175.461	682.993
	HD $n_0 = 550, 500$	$m_a = 2$	min	194.215	299.484	1028.013
			mean	200.883	301.815	1033.423
			max	216.664	305.062	1042.393
		$m_a = \frac{n_0}{5}$	min	269.420	400.387	1588.339
			mean	296.869	415.907	1602.329
			max	308.339	420.444	1609.812
WS	LD $k = 40$	$p = 0.1$	min	39.368	72.182	261.701
			mean	41.943	73.133	264.127
			max	42.929	79.101	266.524
		$p = 0.25$	min	46.982	75.351	294.133
			mean	48.999	80.485	295.321
			max	52.635	95.813	296.284
	HD $k = 300$	$p = 0.1$	min	235.383	373.441	1389.815
			mean	241.482	379.434	1406.931
			max	248.160	395.621	1502.151
		$p = 0.25$	min	257.817	389.898	1543.328
			mean	266.981	401.377	1558.333
			max	277.254	419.524	1652.021

for all nodal and network measures after executing the EAGL Algorithm was 0.72 seconds in NetworkX and 0.12 seconds in MATLAB.

Although the total calculation times for using NetworkX without first implementing the EAGL Algorithm are faster for the unweighted network instances tested, the benefit of preprocessing the geodesic and dependency matrices for rapid nodal and network measure computation in fractions of a second remains invaluable for applications which require it. Consider deliberate planning. By preprocessing the geodesic matrix, alternate interdiction strategies may be considered by utilizing the rapid measure calculations, rather than computing the geodesics to compute measures for every alternative.

3.3.3 Expanding All Geodesics Information Summary.

This section demonstrates that the preprocessing of network information can be extended to include features from the network to compute desired measures in fractions of a second. By processing the EAGL Algorithm in advance, the nodal and network measures for which the necessary information is stored are computed very quickly. The ability to compute the measures rapidly warrants the additional completion time for unweighted networks. The speed with which the measures are computed based on stored geodesic and dependency information allows the modification of their matrix representations to quickly ascertain the impact of node removal from the network to the measures of interest.

3.4 Geodesics After Node Destruction

The previous section demonstrated that nodal and network measure can be computed rapidly when the EAGL Algorithm is implemented prior to their computation. This section utilizes the output of the EAGL Algorithm to update the geodesic lengths as the result of removing each node in the network.

3.4.1 GAND Approach.

After computing and storing the geodesic information for all node pairs using the EAGL Algorithm, an approach that accounts for the Geodesics After Node Destruction (GAND) is developed. The GAND Approach is stated in Algorithm 8 and approximates the impact on the geodesic lengths between all node pairs in the network as a result of each node's destruction. Rather than recompute the geodesic length between all node pairs, the approach examines only the nodes for which their individual destruction will change the length of the geodesic. The only nodes that, if destroyed, result in a larger geodesic length are those nodes that have a single geodesic for a given node pair. Thus, the approach computes the change in the geodesic length as a result of destroying each node that is on the geodesic.

A key feature is that the only input required is the set of geodesic lengths from the EAGL Algorithm. No additional information is required by the approach to update geodesic lengths. This assumption allows for fewer calculations but does not necessarily attain optimal geodesic length solution in certain instances that arise as a result of removing each node. However, the GAND Approach identifies very good solutions rapidly, and a relatively simple check may be utilized to ensure the GAND solution is appropriate prior to its use.

The outputs of the EAGL Algorithm, the length and number of geodesics as well as the predecessors for each node pair in the network, are sufficient to determine if an alternate geodesic is available that does not contain a specified node in most cases. The number of geodesics from node i to node j that pass through node k is given by [10]

$$n_{ij}(k) = \begin{cases} 0 & \text{if } g_{ij} < g_{ik} + g_{kj}, \\ n_{ik} * n_{kj} & \text{otherwise,} \end{cases} \tag{3.2}$$

where g_{ij} and n_{ij} are entries of the geodesic matrix and geodesic counts that are outputs of the EAGL Algorithm, respectively. Given the quantity $n_{ij}(k)$, it is possible to determine

whether node k, which is to be removed from the network, lies on the geodesic from node i to node j. If the total number of geodesics between node pairs is not equal to the number of geodesics containing the specified node, *i.e.*, $n_{ij} \neq n_{ij}(k)$, there exists an alternate geodesic for the given node pair that does not contain node k. Note that the total number of geodesics for a given node pair will never be less than the number of geodesics between the same node pair that contain the specified node. Thus, it can quickly be determined whether there is an alternate geodesic for a given node's removal by utilizing the relationship described by Equation (3.2) to identify an alternate geodesic that does not contain node k.

The GAND Approach requires a $n \times n \times n$ structure to store the change in the length of the geodesic between all node pairs for a given node's destruction. The output is denoted as G^{Δ}, where the entry g^{Δ}_{ijk} denotes the change in the length of the (i, j)-geodesic as a result of destroying node k. As input, the GAND Approach uses the output of the EAGL Algorithm. The GAND Approach iterates through every node pair. For each node pair, each node that appears on the geodesic between them is considered for removal from the network. The approach then determines an alternate geodesic for the node pair. If no alternative exists, the change in the shortest path length is labeled ∞, meaning that the removal of the node severs any possible geodesic between the nodes. The value of ∞ is selected to indicate that there exists no alternate geodesic that will allow a finite path length between the nodes of interest. If an alternate geodesic is identified, the change in geodesic length is updated. The increase in the length of the (i, j)-geodesic can be computed by subtracting the pre-destruction (i, j)-geodesic length from the alternate (i, k)- and (k, j)-geodesic path lengths. This difference is always positive since the path length will only increase after a node on the geodesic is destroyed.

Consider the network depicted in Figure 10, where the length of each arc is one unit. For illustrative purposes, consider only the node pair $(i, j) = (1, 7)$. The length of the geodesic from node 1 to node 7 is 4 units, which can be calculated using the EAGL Algorithm. Using

Algorithm 8 Geodesics After Node Destruction (GAND) Approach

Input

Directed network with nodes $V = \{1, \ldots, n\}$ and arcs $E = (i, j), i, j \in V$.
EAGL Algorithm Outputs (*i.e.*, G, M, N, and P).

Data

K: Queue of nodes contained on current geodesic.

Output

G^{Δ}: $n \times n \times n$ matrix of the change in geodesics between node pairs as a result of removing each node, individually.

Approach

Initialization

$$\text{For } k \in V,\ g^{\Delta}_{ijk} = \begin{cases} -\infty, & \text{if } i = k \text{ and } j = k, \\ \infty, & \text{if } i = k \text{ or } j = k, \\ 0, & \text{otherwise.} \end{cases}$$

Update Geodesics

For $i \in V$,
  For $j \in V$ and $j \neq i$
    Assign node j to K
    While K not empty
      Pop $v \leftarrow K$
      Assign predecessor nodes of v to K, if not already assigned
      If $v \neq i$ and $v \neq j$
        Set $g_{\text{best}} = \infty$
        Identify Alternate Geodesic
          For $k \in V$, $k \neq i, j, v$
            Compute $n_{ik}(v)$
              If $g_{ik} < g_{iv} + g_{vk}$,
                $n_{ik}(v) \leftarrow 0$
              Else,
                $n_{ik}(v) \leftarrow n_{iv} * n_{vk}$
            Compute $n_{kj}(v)$
              If $g_{kj} < g_{kv} + g_{vj}$,
                $n_{kj}(v) \leftarrow 0$
              Else,
                $n_{kj}(v) \leftarrow n_{kv} * n_{vj}$
            Update geodesic length
              If $n_{ik}(v) < n_{ij}$ and $n_{kj}(v) < n_{ij}$
                Update $g_{\text{best}} = \min(g_{\text{best}}, g_{ik} + g_{kj})$.
          Next k.
        $g^{\Delta}_{ijv} = g_{\text{best}} - g_{ij}$.
    Next j
  Next i.

the GAND Approach, each node that appears on a $(1,7)$-geodesic is considered for removal and will comprise the set K. Initially, the set K contains node 7 and, since it is the endpoint of the geodesic in question, is not considered for removal. The predecessor, node 6, is added to set K. Node 6 is then considered for removal. Its predecessor, node 5, is added to K and an alternate geodesic is considered. Because node 6 appears on every possible path from node 1 to node 7, there is no alternate geodesic and $g^{\Delta}_{1,7,6} = \infty$. Node 5 is considered for removal next. There is an alternate $(1,7)$-geodesic (via nodes 3 and 4) that does not contain node 5 and the new geodesic length 1 unit longer than the $(1,7)$ geodesic through node 5, so the change of geodesic length is updated, *i.e.*, $g^{\Delta}_{1,7,5} = 1$. The process is repeated for node 2, with $g^{\Delta}_{1,7,2} = \infty$.

The GAND Approach has the drawback that, when certain network topologies are present, the geodesic information is not sufficient to provide the new geodesic length between a specified node pair for a given node's removal from the network. Two such topology instances, where one is weighted and the other is unweighted, are examined to explain the current shortcoming of the GAND Approach.

Consider the weighted, undirected network depicted in Figure 11 with arc (5,6) having a weight (*i.e.*, length) of 5 units and all other arcs having a weight of 1 unit. When the geodesic matrix is computed for the weighted instance of the network, the $(4,6)$-geodesic has length 3. When node 3 is removed from the network and only geodesic information is

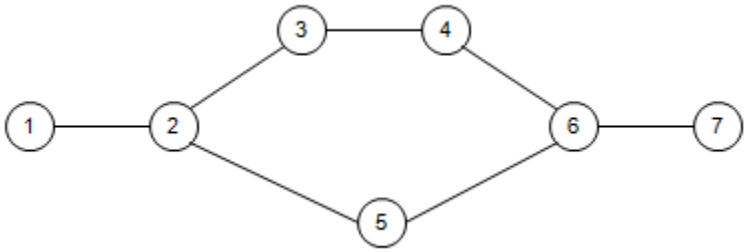


Figure 10. Example Network for GAND Approach

accessible, *i.e.*, the adjacency data is not used so as to improve the speed of the calculation, no alternate $(4, 6)$-geodesic geodesic can be constructed that does not pass through node 3, which is removed, and the GAND Approach gives an infinite geodesic length. Thus, the weighted network requires additional information when unique topologies arise that require solving the geodesic problem (*e.g.*, utilizing Dijkstra's Algorithm), which is what the GAND Approach is circumventing.

However, the GAND Approach is not immune to all unweighted topologies either. Consider the unweighted instance of the network in Figure 12. When considering as input only the geodesic matrix output from the EAGL Algorithm when node 2 is to be removed, for node pair $(1, 3)$, the unique topology of this network instance results in a suboptimal geodesic. The geodesic between node pairs $(1, 3)$, $(1, 5)$,$(1, 6)$, $(5, 3)$, and $(6, 3)$ each have length 2 contain node 2 as the intermediate node along the geodesic. As a result of removing node 2, it is clear that the updated geodesic length is 5 via nodes 4, 5, 6, and 7. However, the geodesic matrix from the EAGL Algorithm does not contain sufficient information to construct this path. Each combination of nodes for an alternate geodesic between nodes 1 and 3 includes node 2. For instance, if node 5 is considered as an alternate intermediate node, the geodesic matrix has two geodesics for the $(1, 5)$ length of 2 units and the single $(5, 3)$-geodesic through node 2 with length 2, which is disallowed as it contains node 2. Similarly, there is no alternate geodesic via node 6. Thus, the GAND Approach identifies an alternate geodesic considering the removal of node 2 via intermediate nodes 8, 9, 10, 11, and 12 with a length of 6, which

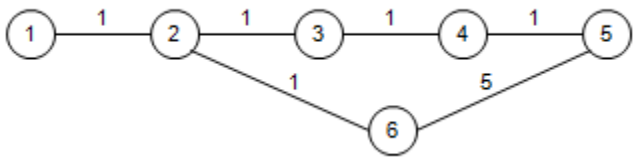


Figure 11. Weighted Network Instance

is longer than the length 5 geodesic mentioned previously and not discovered using GAND Approach assumptions.

Because the GAND Approach may not always identify an alternate geodesic, a relatively simple procedure can determine whether an infinite geodesic change is valid; the *algebraic connectivity* of a network can indicate whether a network is connected. The algebraic connectivity is equivalent to the second smallest eigenvalue of the Laplacian matrix, which is $L(N) = Q(N) - A(N)$, where $Q(N)$ denotes the diagonal matrix whose entries are the degree of the nodes and $A(N)$ denotes the adjacency matrix of the network N. Fielder proved that a network is not connected if and only if the algebraic connectivity equals zero [27]. Recall that, if a geodesic length is given as infinite, there is no path between the node pairs and the network is not connected. Thus, the algebraic connectivity provides a verification of the GAND Approach result. If the algebraic connectivity of the network instance in which a node is removed invalidates the GAND result, the geodesic can be computed using the EAGL Algorithm after the node removed.

For the weighted, undirected network depicted in Figure 11, recall that the GAND Approach solution for node pair $(3, 6)$ when node 4 is removed is infinite. The algebraic connectivity of the network with node 4 removed is 0.519. Because the value of the algebraic

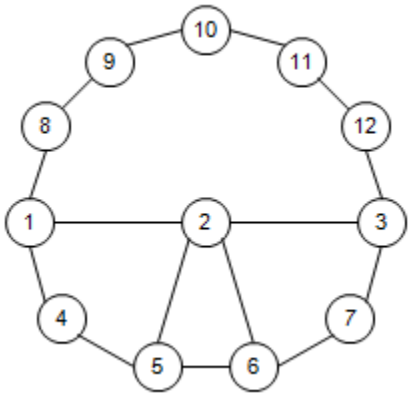


Figure 12. Unweighted Network Instance

connectivity is non-zero, the perturbed network is connected and the infinite geodesic length is invalid.

Unfortunately, the algebraic connectivity validation step for infinite geodesic solutions from the GAND Approach will not identify approach solutions that are finite as was the case in the example network depicted in Figure 12. The GAND Approach solution for node pair $(1, 3)$ when node 2 is removed from the network incorrectly identifies an alternate geodesic having length 6. Because there exists a path between the all node pairs after node 2 is removed, the network is connected and the algebraic connectivity is non-zero. Thus, the validation procedure is only appropriate for infinite GAND solutions.

Of note, the GAND Approach cannot be inserted into the EAGL Algorithm because the change in geodesic length requires geodesic information between all node pairs. The EAGL Algorithm does not have all shortest path information until after its final iteration. It may be possible to collect the values of the change in geodesic lengths recursively and track the destruction impact of nodes as an additional possible extension of the approach.

3.4.2 GAND Extensions.

The GAND Approach may be extended to consider specialized scenarios. Perhaps the source-terminus pairing is fixed. Rather than considering the impact of a node's removal to every possible combination, the approach need only consider the predetermined source-terminus pair. Alternatively, the decision maker may be interested in the impact of destroying a specific node to all possible source-terminus combinations. This also significantly reduces the number of combination required to determine the impact of removal. This section describes such extensions and the implementation modifications of the GAND Approach required.

The GAND Approach can be specialized to consider only a specified source-terminus combination. If an analyst is concerned only with the geodesic between nodes s and t, the

GAND Approach can be modified to output only the impact to the (s,t)-geodesic length. This modification (GAND-st) requires two changes. First, the output is replaced by a $n \times 1$ column vector $G^{st\Delta}$, where entry $g^{st\Delta}(k)$ provides the increase in (s,t)-geodesic length as a result of removing node k from the network. The output is a column vector because the change in geodesic length is always computed with regard to the source-terminus pair, so the output is the impact of each node's removal.

Alternatively, the GAND Approach can be specialized to consider only a single node's removal. Perhaps an analyst is concerned only with the removal or destruction of a specified node's removal from the network. The GAND Approach is modified to determine the increase in the length of the geodesics between all node pairs as a result of the removal of node w from the network. The modification (GAND-w) results in two changes to the approach. The first is replacing the output with a matrix whose size is $n \times n$. The output matrix $G^{w\Delta}$ has entries $g_{ij}^{w\Delta}$, representing the change in geodesic length between nodes i and j as a result of removing node w from the network. The final modification is to examine node v in the GAND Approach only if it is the specified node, i.e., $v = w$. Once node w is encountered on the geodesic, K may be emptied, since node w will not be encountered again.

The GAND Approach modifications can be further combined into a single procedure that considers the increase in geodesic length of a specific (s,t) node pair as a result of removing or destroying a single specified node w. The GAND Approach can be modified as described for each of the previous modifications, resulting in a new approach, GAND-st-w. The resulting output $g^{stw\Delta}$ will be a single value indicating the increase in the length of the (s,t)-geodesic when node w is removed. If the value is infinite, the removal of node w severs the geodesic between node s and node t which must be verified via the aforementioned algebraic connectivity procedure. If the value is 0, there is no change in the geodesic length when node w is removed, *i.e.*, node w is not contained in the specified geodesic.

Preliminary testing was conducted to gauge the impact of the extensions of the GAND Approach. The results suggest that specializing the GAND Approach to a specified source-terminus pair allow the procedure (*i.e.*, GAND-st and GAND-st-w) to complete in less than one second for the 100-node networks tested. When 100-node networks were tested for a single node's removal, the time to complete the GAND-w approach reduced by a factor of five for the preliminary networks tested.

An alternative to the GAND Approach involves using as input a modified EAGL Algorithm that provides as output the k shortest geodesics. Then, the GAND Approach could remove nodes along the shortest geodesic and use each of the remaining $k-1$ shortest geodesics to locate an alternate geodesic that does not include the removed node. This alternative is left for future research and may still have the drawback of network topologies that require more information than is accessible via EAGL Algorithm output data.

3.4.3 New Measures Related to GAND Outputs.

The nodal and network measures related to geodesics can be quickly computed based on the GAND Approach output. Care must be taken to ensure the removal of node k is properly accounted. The nodal measures of eccentricity, total distance, and closeness centrality can be computed with slight modifications using the output of the GAND Approach. The geodesic lengths between all node pairs after node k is removed from the network can be computed with the outputs G, the geodesic matrix from the EAGL Algorithm, and G^{Δ}, the output of the GAND Approach. Since the GAND Approach is concerned with the removal of each node in the network, there are n instances of each nodal measure: one corresponding to each node's removal. Furthermore, each nodal measure has a value for each node in the network. An $n \times n$ matrix of the measure of interest will capture the nodal measure as a result of the removal of each node. Let $e(i,k)$, $\mathrm{td}(i,k)$, and $C_C(i,k)$ respectively represent the eccentricity, total distance, and closeness centrality of node i in the network from which node k has been

Measure	Equation
eccentricity	$e(i,k) = \max(\max\limits_{\substack{j\in V\\ j\neq k}} g_{ijk} + g^{\Delta}_{ijk}, \max\limits_{\substack{j\in V\\ j\neq k}} g_{ijk} + g^{\Delta}_{jik})$ (directed)
	$e(i,k) = \max\limits_{\substack{j\in V\\ j\neq k}} g_{ijk} + g^{\Delta}_{ijk} = \max\limits_{\substack{j\in V\\ j\neq k}} g_{ijk} + g^{\Delta}_{jik}$ (undirected)
total distance (transmission)	$\mathrm{td}(i,k) = \sum\limits_{\substack{j\in V\\ j\neq k}} g_{ijk} + g^{\Delta}_{ijk}$
closeness centrality	$C_C(i,k) = \dfrac{1}{\sum\limits_{\substack{j\in V\\ j\neq k}} g_{ijk} + g^{\Delta}_{ijk}} = \dfrac{1}{\mathrm{td}(i,k)}$

Table 32. Nodal measure updates for GAND Approach output

removed. The modified equations for the calculation of each of these nodal measures is given in Table 32. For each measure, its value when $i = k$ is left empty by construction so it does not skew the network measures that use the value in their calculation.

Similarly, the network measures related to geodesics must account for the removal of each node. Each network measure has a value for each network from which node k is removed. Thus, the network measures will be $n \times 1$ column vectors where each entry represents the measure's value after the removal of node k. Let $\mathrm{diam}(k)$, $\mathrm{rad}(k)$, $P(k)$, $C(k)$, $M(k)$, $\mathrm{w}(k)$, and $\overline{\mathrm{w}}(k)$ denote the respective network measures diameter, radius, peripheral nodes, central nodes, medial nodes, Wiener index, and average distance after node k is removed. The modified equation for each network measure based on the respective nodal measures is given in Table 33. The calculations for network measures do not skip the values for the nodal measures for node k since they were left empty and will not contribute to the calculation of the network measure.

A new measure that indicates the collective impact of removing each node to all shortest paths in the network is the removal index. The removal index $\mathrm{ri}(k)$ can be determined using the output of the GAND Approach because it contains information related to the removal

Measure	Equation
diameter	$\mathrm{diam}(k) = \max_{i \in V} e(i,k)$
radius	$\mathrm{rad}(k) = \min_{i \in V} e(i,k)$
peripheral nodes	$P(k) = \{i \mid e(i,k) = \mathrm{diam}(k)\}$
central nodes	$C(k) = \{i \mid e(i,k) = \mathrm{rad}(k)\}$
medial nodes	$M(k) = \{i \mid \mathrm{td}(i,k) = \min_{j \in V} \mathrm{td}(j,k)\}$
Wiener index (transmission)	$\mathrm{w}(k) = \sum_{i,j \in V} g_{ij} = \sum_{i \in V} \mathrm{td}(i,k)$
average distance	$\overline{\mathrm{w}}(k) = \dfrac{\mathrm{w}(k)}{(n-1)(n-2)}$

Table 33. Network measure updates for GAND Approach output

of node k. Essentially, the index computes the total change in geodesic length as a result of removing a node and is given by

$$\mathrm{ri}(k) = \sum_{i \neq k} \sum_{j \neq k} g_{ijk}^{\Delta}, \tag{3.3}$$

The removal index $\mathrm{ri}(k)$ indicates the total impact to all node pairs of removing node k from the network. The larger the value of the index, the larger the impact of a node's removal.

The removal index for each node in the network depicted in Figure 10 is computed using the output matrix G^{Δ} from the GAND Approach. The removal index is $\mathrm{ri}(k) = 0$ for nodes 1 and 7, $\mathrm{ri}(k) = \infty$ for nodes 2 and 6, $\mathrm{ri}(k) = 4$ for nodes 3 and 4, and $\mathrm{ri}(k) = 8$ for node 5. In other words, the total impact of removing either node 2 or 6 will separate the network into at least two components for at least one node pair in the network, whereas the total impact of removing the other nodes is as given by the removal index.

The removal index for any specified source-terminus pair $\mathrm{ri}_{st}(k)$ can be computed utilizing the output os either the GAND Approach or the GAND-st approach as

$$\mathrm{ri}_{st}(k) = \begin{cases} 0, & \text{if } k = s \text{ or } k = t, \\ g^{\Delta}_{stk} = g^{st\Delta}_{k}, & \text{otherwise,} \end{cases} \tag{3.4}$$

when using the respective output information.

The GAND Approach output will also provide insight regarding the most vital node to geodesics. The most vital node in a shortest path problem is the node for which the removal results in the largest increase in the minimal source-terminus path length. The most vital arc or node problem was solved or refined in [16, 17, 47, 52, 67].

The most vital node to geodesics is typically determined based on a specified source-terminus pair. The GAND Approach provides sufficient information to identify the most vital node to geodesics for every node pair in the network provided the GAND solution is optimal, otherwise, the identified vital nodes are a starting point for determining the actual vital nodes. The most vital nodes to geodesics for the network, $\mathrm{vit}(N)$, for all node pairs are those nodes that have the maximum removal index, while the least vital nodes, $\mathrm{lv}(N)$, for all node pairs in the network are those with the minimum removal index,

$$\mathrm{vit}(N) = \{i | \mathrm{ri}(i) = \max_{j \in V} \mathrm{ri}(j)\}, \tag{3.5}$$

$$\mathrm{lv}(N) = \{i | \mathrm{ri}(i) = \min_{j \in V} \mathrm{ri}(j)\}. \tag{3.6}$$

The most vital nodes in the network are those that should be selected when identifying nodes for which the removal will have the greatest impact regardless of the node pair considered. In other words, the most vital nodes are those nodes that should be targeted when determining attack strategies or hardened when creating defense strategies. The least vital nodes in the

network are those nodes that will have the smallest impact (typically zero) when removed from the network across all node pairs. For a specified source-terminus node pair, the most vital node can be determined as well. For attack strategies requiring high precision with few collateral or cascading effects (*i.e.*, surgical strikes), targets may be identified by the least vital (non-zero) node. To identify such nodes, the definition of the least vital node ((3.6)) would be altered to consider nodes with non-zero removal indices, *i.e.*, $\min_{j \in V} \mathrm{ri}(j) : \mathrm{ri}(j) > 0$.

The most vital nodes for a given node pair can be computed in a manner similar to Equation (3.5) using $\mathrm{ri}_{st}(k)$ instead. Alternatively, the GAND-st approach is suited to this because the construction of the GAND-st approach is such that it outputs the removal index. Thus, the most vital node to the (s,t) geodesic is the node associated with the maximal $g^{st\Delta}$,

$$\mathrm{vit}_{st}(N) = \{i | \mathrm{ri}(i) = \max_{j \in V} \mathrm{ri}_{st}(j) = \max_{j \in V} g_j^{st\Delta}\}. \tag{3.7}$$

Both the most vital node problem and the output of the GAND Approach identify a single target node or a set of nodes having equivalent utility. If the budget or rules of engagement dictate that a single target be attacked, these node identification methods are suitable for determining an attack strategy. However, when the number of nodes in the set of most vital nodes is insufficient to identify a target set of more than one node, these methods fall short. The second most vital node to the geodesic or the node with the second largest removal index combined with the removal of the first node may not have the same effect as the two nodes when chosen as a set. Addressing the shortcoming of this myopic approach will be an avenue of future research. One possible extension occurs when there are predetermined strike packages. The rudimentary approach of removing the nodes in each target set and evaluating the residual network of each is likely the most efficient approach. Identification of target sets using specially tailored implementations of the GAND Approach may be possible.

3.4.4 GAND Testing.

The GAND Approach and its modifications should be compared to an appropriate baseline. The baseline testing verifies that the impact of each node's removal is updated correctly. Additionally, it demonstrates the computational effectiveness of the GAND Approach over the more rudimentary implementation of iteratively removing each individual node and applying the EAGL Algorithm.

This testing is conducted utilizing the same randomly-generated undirected and unweighted network structures that were employed in the previous sections: Erdős-Rényi networks, Barabási-Albert networks, and Watts-Strogatz networks. The parameter setting for the random networks used in testing the GAND Approach and its modifications are listed in Table 18. For each of three types of network structures, 30 replicates of 100-node network instances are generated using the parameters indicated. Half of the test networks have density levels γ less than 0.15 (low-density) and the remainder have densities larger than 0.30 (higher-density). When a weighted network instance is utilized, the distance-weights are selected from a discrete (integer) uniform distribution having a range between 1 and 10, inclusively.

3.4.4.1 100-Node Testing Results.

The actual impact to geodesic lengths is determined by implementing the EAGL Algorithm having as input a modified adjacency matrix with a single node removed. The geodesic lengths between all node pairs are retained for comparison. The EAGL Algorithm is repeated for each of the 100 nodes in the network instance being tested. The minimum, average, and maxim total completion times for the repeated EAGL Algorithm, denoted as "rEAGL", are reported for each of the test instances. In addition, the GAND Approach is implemented after the EAGL Algorithm is performed on the unperturbed network utilizing an unmodified instance of the adjacency matrix. The minimum, average, and maximum

computation times to complete the GAND Approach are recorded, and the EAGL Algorithm and GAND Approach completion times are reported separately. The rEAGL procedure and GAND Approach tests are performed in both MATLAB and Python. The results are further analyzed to assess the accuracy of the reported geodesic lengths using the fast GAND Approach as opposed to the rEAGL procedure. The computation times are given in seconds when computed using Python version 2.7, NetworkX version 1.9.1, and MATLAB version 2012a on an HP Compaq 6005 Pro having a 2.70 GHz AMD Athlon II X2 215 processor and 4.0 GB of RAM. The results of the of the rEAGL procedure and the GAND Approach tests for weighted and unweighted network instances utilizing MATLAB are listed in Tables 34 and 35, respectively. The results for the same network instances when implemented using Python are reported in respective Tables 36 and 37.

The preprocessing step involved in providing input data for the GAND Approach is completed using the EAGL Algorithm. As expected, the single iteration of the EAGL Algorithm required to provide input to the GAND Approach takes approximately one hundredth the time to complete one EAGL iteration for each of the 100 nodes in the network. The NetworkX implementation of the GAND Approach results in determining the impact of each node's removal in less time on average for all network structures except the lower-density instances of the WS structure for the networks tested. Across all the higher-density test instances, the GAND Approach is much faster than the rEASP test for the instances tested.

The nodal and network measure computation times are of paramount importance since the EAGL Algorithm and GAND Approach are preprocessed. The nodal and network measures for weighted network instances when implemented in MATLAB had a total average computation time of 0.18 seconds and 0.12 seconds, respectively. The Python implementation had nodal and network measure calculation times of 2.04 seconds, on average. (There was no discernable difference between the weighted and unweighted network instances.) For the unweighted measure computation times when utilizing MATLAB, there appears to be a

Table 34. Computation times for weighted, 100-node random networks for GAND Approach testing in MATLAB

				rEAGL	GAND	
					EAGL	GAND
ER	LD $p = 0.1$	$\beta = 0$	min	89.385	0.912	17.251
			mean	92.464	0.944	18.598
			max	95.195	0.944	18.598
		$\beta = 1$	min	88.880	0.911	21.609
			mean	94.456	0.964	25.022
			max	102.453	0.964	25.022
	HD $p = 0.5$	$\beta = 0$	min	245.458	2.527	22.004
			mean	254.064	2.602	23.435
			max	281.817	2.602	23.435
		$\beta = 1$	min	245.705	2.511	22.517
			mean	254.485	2.632	23.958
			max	269.248	2.632	23.958
BA	LD $n_0 = 25$	$m_a = 2$	min	85.287	0.863	17.666
			mean	86.868	0.879	19.390
			max	87.934	0.879	19.390
		$m_a = \frac{n_0}{5}$	min	106.613	1.086	18.397
			mean	109.017	1.110	20.257
			max	114.074	1.110	20.257
	HD $n_0 = 55, 50$	$m_a = 2$	min	173.878	1.775	21.910
			mean	176.606	1.810	24.366
			max	187.781	1.810	24.366
		$m_a = \frac{n_0}{5}$	min	0.130	0.130	0.130
			mean	190.893	1.948	23.554
			max	192.008	8.548	11.847
WS	LD $k = 4$	$p = 0.1$	min	63.637	0.648	23.631
			mean	64.291	0.657	28.057
			max	64.864	0.657	28.057
		$p = 0.25$	min	63.828	0.648	20.910
			mean	64.383	0.658	22.776
			max	64.780	0.658	22.776
	HD $k = 30$	$p = 0.1$	min	173.231	1.773	22.760
			mean	174.093	1.788	23.944
			max	175.598	1.788	23.944
		$p = 0.25$	min	173.122	1.772	20.684
			mean	174.155	1.791	22.130
			max	175.260	1.791	22.130

Table 35. Computation times for unweighted, 100-node random networks for GAND Approach testing in MATLAB

				rEAGL	GAND	
					EAGL	GAND
ER	LD $p = 0.1$	$\beta = 0$	min	88.765	0.909	28.498
			mean	91.395	0.936	29.967
			max	93.340	0.936	29.967
		$\beta = 1$	min	86.376	0.885	36.083
			mean	89.217	0.913	40.420
			max	91.756	0.913	40.420
	HD $p = 0.5$	$\beta = 0$	min	297.36	3.062	72.122
			mean	302.091	3.107	73.558
			max	305.769	3.107	73.558
		$\beta = 1$	min	295.265	3.033	70.614
			mean	300.406	3.098	72.659
			max	305.319	3.098	72.659
BA	LD $n_0 = 25$	$m_a = 2$	min	78.227	0.802	16.602
			mean	81.240	0.835	18.739
			max	91.353	0.835	18.739
		$m_a = \frac{n_0}{5}$	min	104.701	1.071	33.024
			mean	105.089	1.078	34.852
			max	106.190	1.078	34.852
	HD $n_0 = 55, 50$	$m_a = 2$	min	143.734	1.472	11.491
			mean	146.571	1.505	12.494
			max	158.010	1.505	12.494
		$m_a = \frac{n_0}{5}$	min	0.122	0.122	0.122
			mean	190.660	1.953	37.345
			max	211.797	13.142	18.737
WS	LD $k = 4$	$p = 0.1$	min	61.173	0.625	32.472
			mean	61.609	0.630	39.908
			max	62.490	0.630	39.908
		$p = 0.25$	min	61.291	0.625	28.953
			mean	61.701	0.631	30.617
			max	62.107	0.631	30.617
	HD $k = 30$	$p = 0.1$	min	167.555	1.715	30.616
			mean	171.210	1.760	39.136
			max	174.284	1.760	39.136
		$p = 0.25$	min	174.393	1.787	30.980
			mean	175.623	1.811	32.223
			max	176.748	1.811	32.223

Table 36. Computation times for weighted, 100-node random networks for GAND Approach testing in Python

				rEAGL	GAND	
					EAGL	GAND
ER	LD $p = 0.1$	$\beta = 0$	min	24.827	0.229	5.519
			mean	25.735	0.266	6.095
			max	27.111	0.266	6.095
		$\beta = 1$	min	24.956	0.224	6.869
			mean	25.644	0.257	7.991
			max	26.274	0.257	7.991
	HD $p = 0.5$	$\beta = 0$	min	66.427	0.614	7.258
			mean	67.109	0.658	7.668
			max	67.804	0.658	7.668
		$\beta = 1$	min	66.045	0.608	7.113
			mean	66.972	0.673	7.695
			max	68.524	0.673	7.695
BA	LD $n_0 = 25$	$m_a = 2$	min	23.969	0.216	5.403
			mean	24.280	0.237	6.126
			max	24.748	0.237	6.126
		$m_a = \frac{n_0}{5}$	min	29.690	0.269	5.771
			mean	30.063	0.298	6.438
			max	30.526	0.298	6.438
	HD $n_0 = 55, 50$	$m_a = 2$	min	47.135	0.433	7.287
			mean	48.009	0.485	7.896
			max	53.498	0.485	7.896
		$m_a = \frac{n_0}{5}$	min	51.169	0.470	7.083
			mean	51.557	0.510	7.815
			max	51.985	0.726	8.607
WS	LD $k = 4$	$p = 0.1$	min	18.020	0.159	7.141
			mean	18.250	0.192	8.849
			max	18.638	0.192	8.849
		$p = 0.25$	min	18.122	0.161	6.613
			mean	18.378	0.187	7.270
			max	18.739	0.187	7.270
	HD $k = 30$	$p = 0.1$	min	46.574	0.431	7.337
			mean	47.000	0.460	7.852
			max	48.004	0.460	7.852
		$p = 0.25$	min	46.333	0.427	6.702
			mean	47.237	0.470	7.277
			max	47.669	0.470	7.277

Table 37. Computation times for unweighted, 100-node random networks for GAND Approach testing in Python

				rEAGL	GAND	
					EAGL	GAND
ER	LD $p = 0.1$	$\beta = 0$	min	19.450	0.174	9.720
			mean	20.559	0.202	10.549
			max	22.362	0.202	10.549
		$\beta = 1$	min	19.091	0.172	13.522
			mean	20.165	0.206	16.353
			max	21.798	0.206	16.353
	HD $p = 0.5$	$\beta = 0$	min	66.217	0.626	24.285
			mean	67.222	0.647	24.854
			max	68.205	0.647	24.854
		$\beta = 1$	min	65.603	0.625	24.058
			mean	68.297	0.695	25.039
			max	70.996	0.695	25.039
BA	LD $n_0 = 25$	$m_a = 2$	min	17.671	0.155	6.075
			mean	18.072	0.179	6.843
			max	19.567	0.179	6.843
		$m_a = \frac{n_0}{5}$	min	23.612	0.213	13.415
			mean	24.173	0.238	14.274
			max	26.052	0.238	14.274
	HD $n_0 = 55, 50$	$m_a = 2$	min	37.808	0.345	4.223
			mean	38.654	0.374	4.647
			max	40.231	0.374	4.647
		$m_a = \frac{n_0}{5}$	min	44.361	0.410	13.506
			mean	45.911	0.448	14.518
			max	49.643	0.608	15.756
WS	LD $k = 4$	$p = 0.1$	min	13.294	0.113	12.383
			mean	14.291	0.151	16.283
			max	15.021	0.151	16.283
		$p = 0.25$	min	13.308	0.114	10.081
			mean	13.692	0.138	10.882
			max	15.608	0.138	10.882
	HD $k = 30$	$p = 0.1$	min	40.442	0.373	11.209
			mean	42.256	0.406	14.507
			max	46.669	0.406	14.507
		$p = 0.25$	min	41.104	0.382	10.141
			mean	42.213	0.413	10.767
			max	44.854	0.413	10.767

correlation between network density and average computation time ($r^2 = 0.76$) with lower-density networks requiring less time to compute the measures. The weighted MATLAB implementation and both Python implementations had no correlation between network density and computation time ($r^2 < 0.07$). The total completion times for the GAND Approach are faster when implemented utilizing Python; however, because the GAND Approach is processed in advance, the measure calculation times indicate that the MATLAB implementation is faster.

The fast computation times of the GAND Approach demonstrate that it should be implemented, provided the accuracy of the geodesics after a node's removal is high. The total number of differences, n_δ, is computed for the output of the rEAGL procedure and the GAND Approach, *i.e.*, the number of instances for which the geodesic length after a node's removal is not equal in the two methods. In addition, the total difference, δ_T, between the geodesic lengths in these instances is calculated, by subtracting the resultant GAND geodesic length from the rEAGL geodesic length. The average difference $\overline{\delta}_T$ is the ratio of the sum of the differences to the total number of differences for a network instance,

$$\overline{\delta_T} = \begin{cases} 0, & \text{if } n_\delta = 0, \\ \dfrac{\delta_T}{n_\delta}, & \text{otherwise.} \end{cases} \tag{3.8}$$

The accuracy, a, is computed by computing the ratio of the total number of differences to the total number of geodesic lengths computed,

$$a = 1 - \frac{n_\delta}{n^3}. \tag{3.9}$$

The maximum, average, and minimum of each accuracy measure (*i.e.*, number of differences, total difference, average difference, and accuracy) for each of the weighted and unweighted network instances tested are reported in Tables 38 and 39, respectively. The accuracy in

the tables is given as an integer value of 1 if there are no differences between the rEAGL procedure and the GAND Approach across all replicates.

In the accuracy tables, it is apparent that the GAND Approach is less accurate for lower-density network instances tested. The number of infinite differences between the actual geodesic lengths and the GAND geodesic lengths appear more frequently in the lower-density networks. The weighted network instances result in more infinite-length differences, but only the higher-density instances for the BA structure exhibit such infinite differences. Furthermore, the average difference for the higher-density network instances was much smaller. The unweighted network instances also demonstrated difficulty handling the lower densities with infinite lengths appearing only in the LD rows of Table 39. The lower-density networks have fewer arcs between the nodes, so there are fewer geodesic from which to select an alternative. Coupled with the GAND Approach limitations stemming from EAGL input assumptions, the lower-density network instances cause more discrepancies for the instances tested.

Table 40 provides the number of instances tested that the removal of a node resulted in the GAND Approach claiming the network was disconnected when it was not. The BA network structure had the largest number of incorrect infinite geodesic updates. This result is not surprising since the nodes not included in the initial, fully connected group are sparsely connected and would require a longer path to reach the initial group again, which is exacerbated in the weighted network instances. The results confirm the observation that the lower-density, weighted network structures provide challenges for the GAND Approach because there are fewer arcs over which alternate geodesics can be identified.

3.4.5 Geodesics After Node Destruction Summary.

This section demonstrated the implementation of an approach to assess the impact of the removal of each node in a network. The GAND Approach has the shortcoming that sufficient geodesic information is not always available from the EAGL Algorithm inputs to generate

Table 38. Accuracy measures for weighted, 100-node random networks for GAND Approach

				n_δ	δ_T	$\overline{\delta_T}$	a
ER	LD $p = 0.1$	$\beta = 0$	min	774	1316	1.567	0.998
			mean	1165	∞	∞	0.999
			max	1796	∞	∞	0.999
		$\beta = 1$	min	1424	2870	1.653	0.995
			mean	2335	∞	∞	0.998
			max	4834	∞	∞	0.998
	HD $p = 0.5$	$\beta = 0$	min	306	326	1.040	1.000
			mean	368	400	1.087	1.000
			max	438	488	1.129	1.000
		$\beta = 1$	min	272	296	1.052	1.000
			mean	361	393	1.090	1.000
			max	450	490	1.122	1.000
BA	LD $n_0 = 25$	$m_a = 2$	min	4214	∞	∞	0.991
			mean	6342	∞	∞	0.994
			max	8924	∞	∞	0.994
		$m_a = \frac{n_0}{5}$	min	1438	∞	∞	0.997
			mean	2372	∞	∞	0.998
			max	3368	∞	∞	0.998
	HD $n_0 = 55, 50$	$m_a = 2$	min	3316	∞	∞	0.994
			mean	4736	∞	∞	0.995
			max	6408	∞	∞	0.995
		$m_a = \frac{n_0}{5}$	min	382	470	1.167	0.999
			mean	660	∞	∞	0.999
			max	1276	∞	∞	1.000
WS	LD $k = 4$	$p = 0.1$	min	2836	∞	∞	0.980
			mean	6628	∞	∞	0.993
			max	19866	∞	∞	0.993
		$p = 0.25$	min	1446	8732	4.366	0.994
			mean	2721	∞	∞	0.997
			max	5788	∞	∞	0.997
	HD $k = 30$	$p = 0.1$	min	318	388	1.129	0.999
			mean	425	509	1.198	1.000
			max	634	786	1.257	1.000
		$p = 0.25$	min	324	386	1.142	0.999
			mean	426	512	1.200	1.000
			max	520	616	1.269	1.000

Table 39. Accuracy measures for unweighted, 100-node random networks for GAND Approach

				n_δ	δ_T	$\overline{\delta_T}$	a
ER	LD $p = 0.1$	$\beta = 0$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1
		$\beta = 1$	min	0	0	0	1
			mean	1.6	∞	∞	1.000
			max	36	∞	∞	1.000
	HD $p = 0.5$	$\beta = 0$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1
		$\beta = 1$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1
BA	LD $n_0 = 25$	$m_a = 2$	min	0	0	0	1
			mean	7.9	∞	∞	1.000
			max	22	∞	∞	1.000
		$m_a = \frac{n_0}{5}$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1
	HD $n_0 = 55, 50$	$m_a = 2$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1
		$m_a = \frac{n_0}{5}$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1
WS	LD $k = 4$	$p = 0.1$	min	2	2	1	0.999
			mean	63.5	∞	∞	1.000
			max	722	∞	∞	1.000
		$p = 0.25$	min	0	0	0	1
			mean	4.9	5.5	0.9	1.000
			max	18	20	2	1.000
	HD $k = 30$	$p = 0.1$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1
		$p = 0.25$	min	0	0	0	1
			mean	0	0	0	1
			max	0	0	0	1

Table 40. Number of instances of invalid infinite geodesic for 100-node random networks for GAND Approach

	Density	Parameter	Unweighted	Weighted
ER	LD	$\beta = 0$	0	11
	$p = 0.1$	$\beta = 1$	1	60
	HD	$\beta = 0$	0	0
	$p = 0.5$	$\beta = 1$	0	0
BA	LD	$m_a = 2$	41	595
	$n_0 = 25$	$m_a = \frac{n_0}{5}$	0	256
	HD	$m_a = 2$	0	587
	$n_0 = 55, 50$	$m_a = \frac{n_0}{5}$	0	44
WS	LD	$p = 0.1$	1	134
	$k = 4$	$p = 0.25$	0	73
	HD	$p = 0.1$	0	0
	$k = 30$	$p = 0.25$	0	0

the optimal updated geodesic length for some network topology instances. This limitation is overcome by utilizing the algebraic connectivity measure to verify any solutions indicating the network is disconnected. The MATLAB implementation of the GAND Approach appears to be more effective than the alternative implementations examined since the matrix manipulation (for which MATLAB was created) enables faster nodal and network measure calculations and can rapidly verify when secondary processing is required if the solution is infinite. The GAND Approach provides an effective preprocessing step that yields defined solutions regarding the impact of the removal of each node in the network.

The given implementation of the GAND Approach does not update the number of shortest paths, the node dependencies, or the adjacency matrix. Future extensions of the approach may include updating these values so the nodal and network measures (*e.g.*, degree centrality, betweenness centrality, and clustering coefficients) can be readily computed; the nodal measures that depend only on the lengths of geodesics can be computed for any single node destroyed.

Future Work. When studying the impact of large-scale node removals to networks, it is important to quickly characterize the connectedness of the residual network. Certain measures are computed utilizing the eigenvalues and associated eigenvectors of the Laplacian matrix, which are particularly useful when considering large-scale node removal sets. Recall that the Laplacian matrix $L(N)$ is defined as $L(N) = Q(N) - A(N)$, where $Q(N)$ denotes the diagonal matrix whose entries are the degree of the nodes and $A(N)$ denotes the adjacency matrix of the network N. The number of components and their respective sizes within the residual network after the large-scale node removal are computed using the eigenvalues and associated eigenvectors of the Laplacian matrix of the residual network, $L(N')$, where N' is the network with the large-scale node set removed.

The number of eigenvalues of $L(N')$ having value 0 indicates the number of components in the network [48]. Furthermore, Ding *et al.* observed that the graph is disconnected between the nodes at which the eigenvector changes value [23]. Therefore, the number of nodes in each of the disconnected components can be found by counting the number of unique values (within epsilon tolerance) of the eigenvector associated with any eigenvalue equal to zero. Furthermore, the smallest non-zero eigenvalue of the Laplacian matrix indicates how well-connected the largest component is; if the value is near zero, there exist few ties connecting potential components. Thus, the eigenvalues of Laplacian matrix for the residual network provide a measure of the impact to the network of the set's removal. This computation is processed quickly in software designed for matrix manipulation such as MATLAB.

This topic is not addressed in the open literature in terms of generating target lists for attacking large portions of a network or assessing the impact to the residual network after a large-scale attack. By applying these measures to the large-scale nodal removal problem, analysts will have the tools to provide large-scale node removal impacts to decision makers with regard to offensive and/or defensive tasks that require such information.

3.5 Summary

This chapter provides an array of approaches that allow an analyst to inform decision makers which nodes to target for the largest impact on the network in question or for the least impact. It demonstrated that measure computation time can be greatly reduced when geodesic information is retained, that extending the information stored allows the calculation of network measures just as quickly, and that utilizing this information allows the assessment of each node's removal from the network. Without understanding the relationship between geodesics and the measures related to geodesic lengths, characterizing the importance of nodes within the network using nodal and/or network measures in a timely manner is prohibitive.

In addition, the network information that is preprocessed can be extended to compute additional measures in fractions of a second. The EAGL Algorithm compiles and stores the necessary information required for the rapid computation of nodal and network measures. The speed with which the measures are computed based on stored geodesic and dependency information allows the modification of their matrix representations to quickly ascertain the impact of node removal from the network to the measures of interest. The GAND Approach provides a fast preprocessing step that yields good solutions regarding the impact of the removal of each node in the network. These contributions are summarized in Table 41.

Table 41. Destroying Interdiction Task Contributions

Interdiction Task	Section	Contribution	Description
Destroy	3.3.1	EAGL Algorithm	The Extended All Geodesics Lengths (EAGL) Algorithm takes the features of previous algorithms (*i.e.*, retaining all geodesic and predecessor information, computing node dependencies, and counting occurrences of nodes on geodesics) to construct a more robust algorithm.
	3.4.1	GAND Approach	The Geodesics After Node Destruction (GAND) Approach determines the impact on the geodesic lengths between all node pairs in the network as a result of each node's destruction.

IV. Destroy Interdiction Tasks

4.1 Introduction

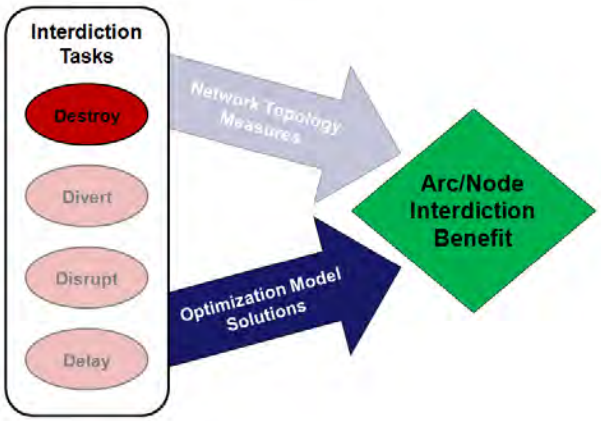


Figure 13. Research Framework: Modeling Destroy Interdiction Tasks

This chapter develops a mathematical programming modeling framework for network interdiction via the destruction of nodes. Thus, the research addresses the 'destroy' interdiction task in the models approach of the research framework as depicted in Figure 13. The models developed in this chapter extend typically utilized methods for arc interdiction. The shortest path arc interdiction is a new approach for destroying arcs when also considering the shortest paths of the network operator. The chapter describes an array of optimization models for destroying arcs in a network while considering the network's maximum flow or shortest path(s).

Operational planners may choose to utilize interdiction tasks to target arcs. In the physical sense, arcs may represent routes or points at which several infrastructures are located in close proximity. Within social networks, arcs may represent ties between people that may be targeted given the correct situation.

4.2 Maximum Flow Models

The network interdiction model of Wood [68] is a bilevel programming model which utilizes a minimum capacity cut problem to determine an optimal arc cut set for the leader, subject to a resource constraint that, if restrictive enough, allows some flow through the network, which is minimized in the objective. The follower's maximum flow problem is solved implicitly. Consider a cut that is based on cost and is subsequently resource constrained in such a way as to allow residual flow for the follower. The follower's maximum flow problem is no longer solved implicitly. Rather, decision variables indicating the flow across the nodes in the network are required to explicitly determine the follower's solution.

The standard minimum cut model does not require consideration for residual network flow because the solution is always a cut set through which no flow is allowed. The flow objective is included in the arc destroying model to illustrate its inclusion, confirm that flow is interdicted, and as a reference for subsequent models. The formulation for the maximum

flow arc destroying problem (**MAD**) follows:

$$\min \quad \sum_{(i,j)\in E} c_{ij}y_{ij} - \lambda x_{ts} \tag{4.1a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{4.1b}$$

$$u_s = 0, \tag{4.1c}$$

$$u_t = 1, \tag{4.1d}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = 0, \qquad \forall i \in V, \tag{4.1e}$$

$$x_{ij} \leq b_{ij}(1 - y_{ij}), \qquad \forall (i,j) \in E, \tag{4.1f}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{4.1g}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.1h}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{4.1i}$$

The objective (4.1a) ensures that the leader's minimum cost s-t cut set is selected while maximizing the follower's remaining source-to-terminus flow on the network, which will result in $x_{ts} = 0$, and is included as a placeholder to inform subsequent models. The cost to interdict arc (i,j) is represented by c_{ij}. The value of λ determines which solution methodology the solver prefers. Larger values of λ cause the solver to utilize methods for maximum flow solutions while smaller values of λ direct the optimization software to prefer minimum cut solution methods.

The leader's problem consists of the first term of the objective function and Constraints (4.1b)-(4.1d) and (4.1h)-(4.1i). Constraint (4.1b) ensures that an s-t cut is identified. The source node is identified in (4.1c). In Constraint (4.1d), the terminus node is assigned $u_t = 1$ to enforce the cut between the source and terminus. Finally, Constraints (4.1h) and (4.1i) ensure the leader's decision variable values are binary.

The follower's problem is the maximum flow problem and consists of the last term in the objective function and Constraints (4.1e)-(4.1g). Constraint (4.1e) ensures flow conservation at each node. The flow on each arc is non-negative (4.1g). Constraint (4.1f) ensures the flow is zero if the leader interdicts the arc; otherwise it is bounded above by the arc's capacity b_{ij}.

Whereas the solutions to the MAD model will always result in zero flow for the follower's residual network, the weights of the leader and follower components of the objective function are important. The value of λ associated with the follower's part of the objective can account for a weighting of the leader's objective. Let λ_1 and λ_2 denote a weighting of the leader and follower objectives, respectively. Then, by dividing both objectives by λ_1, we have a coefficient of 1 for the leader and $\lambda = \dfrac{\lambda_2}{\lambda_1}$ for the given weight in the objective function. To ensure that the leader has preemptive preference over the follower, the weights for λ_1 and λ_2 can be determined by utilizing Sherali's Algorithm 1 [58] where the preemptive weights can be determined based on the upper bounds of the respective objectives and based on the relationships established by Sherali and Soyster [59].

Consider a military battle being fought on three fronts and an adversary using four supply depots from which to deliver military items. Figure 14 depicts this notional military transportation network example as used by Ghare *et al.* [33] and Wood [68]. The capacity and interdiction costs are enumerated in Table 42. When this problem is solved using the MAD model, the solution is to interdict arcs {(2,9), (2,6), (3,6), (7,10), (11,14), (11,15), (8,12), (5,12)} at a total interdiction cost of 34 with a maximum flow for the residual network of 0 units.

As noted, the maximum flow solution for any follower within the MAD model is 0 units of flow because every arc in the cut set is targeted for destruction. The value of λ impacts the solution times for the MAD model. Recall that the value of λ determines which objective function component the solver prioritizes: the leader's minimum cost cut, or the follower's

Arc	Capacity	Cost	Arc	Capacity	Cost
(1,2)	1000	100	(6,10)	60	7
(1,3)	1000	100	(7,10)	120	4
(1,4)	1000	100	(7,11)	150	6
(1,5)	1000	100	(8,11)	120	6
(2,6)	60	5	(8,12)	80	4
(2,9)	70	4	(9,13)	80	4
(2,7)	60	5	(9,14)	50	5
(3,6)	50	3	(10,13)	100	5
(3,7)	50	3	(10,14)	80	4
(3,8)	60	5	(11,14)	180	6
(4,7)	100	3	(11,15)	100	4
(4,8)	80	5	(12,14)	80	5
(5,7)	50	5	(12,15)	100	6
(5,8)	100	5	(13,16)	1000	100
(5,12)	80	4	(14,16)	1000	100
(6,9)	60	4	(15,16)	1000	100

Table 42. Data for the notional military transportation network in Figure 14 [33, 68]

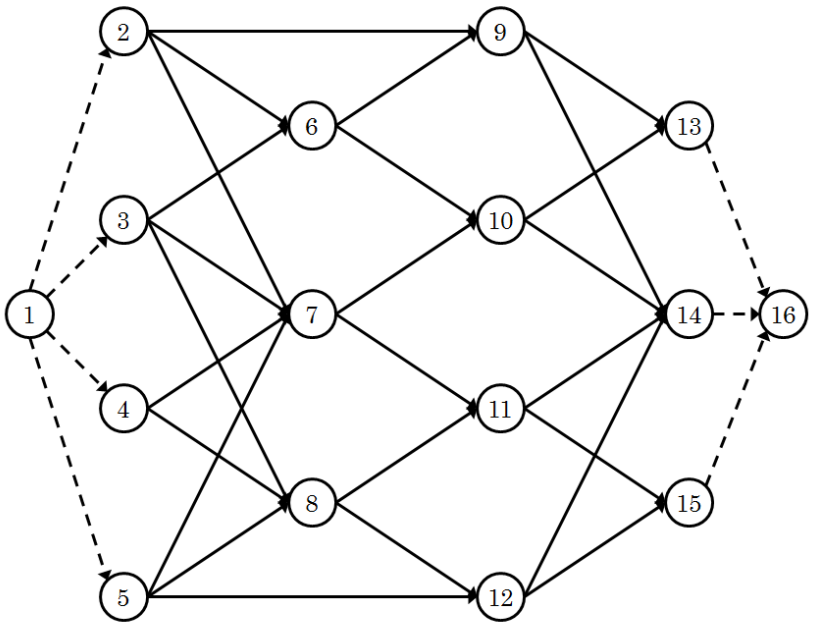


Figure 14. A notional military transportation network [33, 68]

maximum flow. For intermediate values of λ, the solver will attempt to balance a combination of the two objectives, resulting in extended solution times. Table 43 reports the mean solution time and number of iterations for the network instances tested. A two-tailed paired-t test statistic is utilized to determine whether there is a difference in the measure (*i.e.*, mean solution time or number of iterations) with $\lambda = 0.001$ or $\lambda = 0.00001$. Table 44 lists the mean solution time and number of iterations for the network instances tested and the two-tailed paired-t test statistic to determine whether there is a difference with $\lambda = 0.001$ or $\lambda = 1$. In cases for which the p-value is less than 0.05, there is sufficient evidence at the 0.05 level of significance to reject the claim that there is no difference in the measure when the λ-value increases to 1. For larger p-values, there is insufficient evidence at the 0.05 level of significance to reject the claim of no difference in the measure when the λ-value increases to 1.

Examination of the p-values comparing $\lambda = 0.001$ and $\lambda = 0.00001$ for the mean solution time and number of iterations are larger than 0.05 for all network cases tested. There is insufficient evidence at the 0.05 level of significance to reject the claim of no difference in the measure. Therefore, the decision to use either value is arbitrary, and the model will be solved utilizing $\lambda = 0.001$ unless otherwise noted.

Comparison of $\lambda = 0.001$ and $\lambda = 1$ in the MAD model resulted in p-values for the mean solution time and the number of iterations that are smaller than 0.05 for all network types. There is sufficient evidence at the 0.05 level of significance to reject the claim of no difference in the measure for the network instances tested. This result demonstrates the value in ensuring the leader's objective is preferred over the follower's objective to reduce solution times and leverage the solver's efficiency in solving minimum cut problems.

Network density is graphed against solution times in Figure 15 for the network instances tested with $\lambda = 0.001$. The line represents a linear approximation of the relationship between density and solution time. For the networks tested, solution time increases with an associated

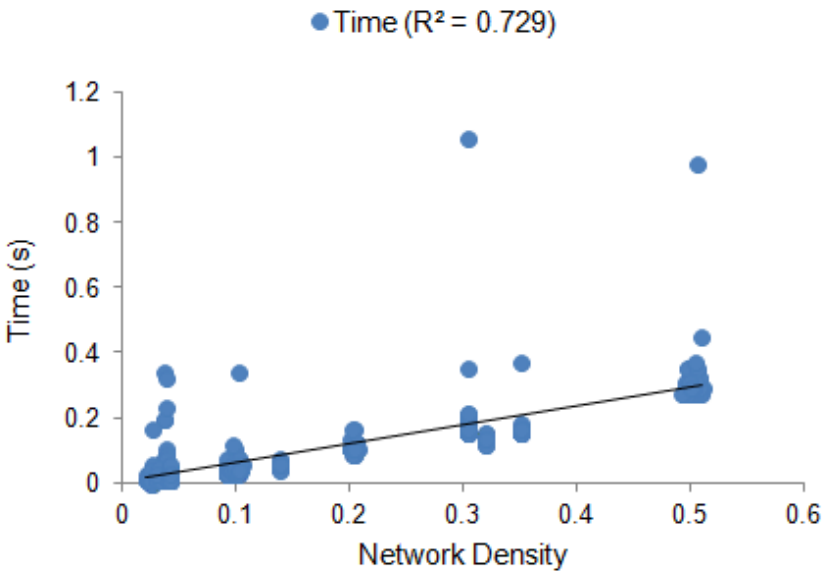


Figure 15. Density vs solution time for 100-node network instances tested

increase in network density. The coefficient of determination R^2 between density and solution time, given in parentheses in the legend of the chart for the tested network cases, is 0.729. The large value of R^2 suggests that there exists a strong linear relationship between density and solution time when taking into account the variance over the 30 replicates of each case. The number of iterations did not exhibit as strong a linear relationship ($R^2 = 0.321$) with network density for the networks instances tested.

Table 43. Comparison of results for MAD with $\lambda = 0.001$ and $\lambda = 0.00001$

Structure	Case	$\lambda = 0.001$			$\lambda = 0.00001$			2-Tail p-value	
		# Solved	Mean Time	Mean Its	# Solved	Mean Time	Mean Its	Time	Iterations
ER	1	30 / 30	0.053	325.7	30 / 30	0.051	320.3	0.540	0.245
	2	30 / 30	0.067	375.5	30 / 30	0.073	380.6	0.624	0.152
	3	30 / 30	0.335	712.9	30 / 30	0.321	714.5	0.570	0.346
	4	30 / 30	0.303	711.4	30 / 30	0.324	709.9	0.080	0.357
BA	5	30 / 30	0.040	33.3	30 / 30	0.040	33.3	0.683	1
	6	30 / 30	0.060	128.9	30 / 30	0.059	128.9	0.717	1
	7	30 / 30	0.140	84.6	30 / 30	0.138	84.6	0.348	1
	8	30 / 30	0.174	259.4	30 / 30	0.178	259.4	0.309	1
WS	9	30 / 30	0.024	260.7	30 / 30	0.026	259.5	0.455	0.694
	10	30 / 30	0.028	239.5	30 / 30	0.024	234.4	0.170	0.275
	11	30 / 30	0.222	767.2	30 / 30	0.200	764.9	0.485	0.745
	12	30 / 30	0.173	637.0	30 / 30	0.178	637.2	0.429	0.969
PNDCG	13	30 / 30	0.018	76.8	30 / 30	0.027	76.5	0.174	0.371
	14	30 / 30	0.024	80.4	30 / 30	0.023	80.1	0.996	0.096
	15	30 / 30	0.019	133.9	30 / 30	0.020	133.1	0.812	0.306
	16	30 / 30	0.018	139.3	30 / 30	0.021	139.3	0.096	1
	17	30 / 30	0.111	516.9	30 / 30	0.115	516.7	0.289	0.953
	18	30 / 30	0.114	508.7	30 / 30	0.110	507.2	0.207	0.53
Grid	19	30 / 30	0.048	490.9	30 / 30	0.035	476.3	0.190	0.206
	20	30 / 30	0.046	533.5	30 / 30	0.043	532.0	0.679	0.866
	21	30 / 30	0.054	356.9	30 / 30	0.042	347.4	0.400	0.116
Star	22	30 / 30	0.026	298.6	30 / 30	0.026	298.1	0.756	0.463
	23	30 / 30	0.020	171.5	30 / 30	0.020	171.5	0.987	1
	24	30 / 30	0.029	371.3	30 / 30	0.025	371.7	0.033	0.916

Table 44. Comparison of results for MAD with $\lambda = 0.001$ and $\lambda = 1$

Structure	Case	$\lambda = 0.001$			$\lambda = 1$			2-Tail p-value	
		# Solved	Mean Time	Mean Its	# Solved	Mean Time	Mean Its	Time	Iterations
ER	1	30 / 30	0.053	325.7	30 / 30	0.178	558.4	< 0.001	< 0.001
	2	30 / 30	0.067	375.5	30 / 30	0.233	892.7	< 0.001	0.034
	3	30 / 30	0.335	712.9	30 / 30	1.669	1210.3	< 0.001	< 0.001
	4	30 / 30	0.303	711.4	30 / 30	1.498	1117.9	< 0.001	< 0.001
BA	5	30 / 30	0.040	33.3	30 / 30	0.052	50.9	0.002	< 0.001
	6	30 / 30	0.060	128.9	30 / 30	0.064	163.2	0.075	0.026
	7	30 / 30	0.140	84.6	30 / 30	0.174	103.9	0.072	0.019
	8	30 / 30	0.174	259.4	30 / 30	0.179	326.5	0.492	< 0.001
WS	9	30 / 30	0.024	260.7	30 / 30	0.275	3099.4	< 0.001	0.002
	10	30 / 30	0.028	239.5	30 / 30	0.158	865.3	< 0.001	0.007
	11	30 / 30	0.222	767.2	30 / 30	0.718	974.1	< 0.001	< 0.001
	12	30 / 30	0.173	637.0	30 / 30	0.713	880.2	< 0.001	< 0.001
PNDCG	13	30 / 30	0.018	76.8	30 / 30	0.023	97.5	0.065	0.012
	14	30 / 30	0.024	80.4	30 / 30	0.025	86.2	0.788	0.017
	15	30 / 30	0.019	133.9	30 / 30	0.078	890.8	0.023	0.175
	16	30 / 30	0.018	139.3	30 / 30	0.052	208.4	< 0.001	< 0.001
	17	30 / 30	0.111	516.9	30 / 30	0.324	692.4	< 0.001	< 0.001
	18	30 / 30	0.114	508.7	30 / 30	0.318	671.3	< 0.001	< 0.001
Grid	19	30 / 30	0.048	490.9	30 / 30	94.241	2534908.4	< 0.001	< 0.001
	20	30 / 30	0.046	533.5	30 / 30	3482.202	89744454.7	< 0.001	< 0.001
	21	30 / 30	0.054	356.9	30 / 30	1.737	31788.8	< 0.001	< 0.001
Star	22	30 / 30	0.026	298.6	30 / 30	0.293	4084.3	0.025	0.048
	23	30 / 30	0.020	171.5	30 / 30	0.127	339.4	< 0.001	< 0.001
	24	30 / 30	0.029	371.3	30 / 30	0.618	9478.4	< 0.001	< 0.001

4.3 Shortest Path Models

Models have been utilized to represent interdicting arcs on the shortest path between a source and a terminus. Golden [35], Fulkerson and Harding [31], and Israeli and Wood [42] formulated similar models that solve the shortest path problem and either destroy an arc or increase the length of a selected arc to maximize the overall length of the shortest path. The model developed in this section considers interdicting the shortest path from a minimum cost cut set perspective. The benefit of considering a minimum cost cut set is that the model can be extended to interdict the k shortest paths.

The leader's portion of the model is similar to the MAD model in that the leader utilizes several decision variables to identify a cut set between the source and terminus. The leader's decision variables identify a single arc that is in the intersection of the minimum cost cut set and the shortest path. This single arc to be targeted for destruction is indicated by $z_{ij} = 1$. The objective is weighted to favor the leader's minimum cut problem, which is solved relatively quickly by optimization software. The remaining decision variables for the leader, y_{ij} and u_i, respectively represent the arcs in the cut set that are not interdicted and the node partition to identify the cut set.

The follower's problem is the shortest path formulation. A single unit of flow is introduced at the source and a demand of one unit is required at the terminus. The follower's decision variable x_{ij} indicates which arcs the single commodity traverses to reach the terminus. By minimizing the sum of the lengths of the arcs traversed, a shortest path is identified.

The cut set is partitioned into the set of nodes that are not interdicted and a single arc that is targeted for destruction. The objective determines the minimum cost cut set which includes a single arc through which the shortest path travels. The cost to interdict arc (i, j) and the distance from node i to node j are represented by c_{ij} and d_{ij}, respectively. The

formulation for the shortest path arc destroying problem (**SAD**) follows:

$$\min \quad \sum_{(i,j)\in E} c_{ij}(y_{ij} + z_{ij}) + \lambda \sum_{(i,j)\in E} d_{ij}x_{ij} \tag{4.2a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} + z_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{4.2b}$$

$$u_s = 0, \tag{4.2c}$$

$$u_t = 1, \tag{4.2d}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V, \tag{4.2e}$$

$$x_{ij} \leq 1 - y_{ij}, \qquad \forall (i,j) \in E, \tag{4.2f}$$

$$\sum_{(i,j)\in E} z_{ij} = 1, \tag{4.2g}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.2h}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.2i}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.2j}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{4.2k}$$

The objective (4.2a) ensures that the leader targets the arc that is contained in the intersection of the minimum cost cut set and the follower's shortest s-t path. The value of λ should be selected to ensure the model prefers the leader's minimum cut solution method. To ensure that the leader has preemptive preference over the follower, the weight for λ can be determined by utilizing Sherali's Algorithm 1 [58].

The leader's (attacker's) problem consists of the first term of the objective function and constraints (4.2b)-(4.2d), which are as described for the MAD model.

The follower is concerned with the shortest path problem, consisting of the last term in the objective function and constraints (4.2e)-(4.2f). Constraints (4.2e) ensure flow conservation at each node, a single unit of flow originating from the source, and a single unit of flow terminating at the destination. Constraint (4.2f) ensures the arc is not on the shortest path if the leader interdicts the arc. Notice that a single arc (indicated by the non-zero z_{ij} variable) is contained in the cut set and is on the shortest path.

Consider again the notional military transportation network example given in Figure 14 and Table 42. When this problem is solved using the SAD model, the arc that should be targeted for interdiction is arc (2,9) at a cost of 4 units. The minimum cost cut set contains arcs {(2,9), (2,6), (3,6), (7,10), (11,14), (11,15), (8,12), (5,12)} with a total interdiction cost of 34 units. The shortest path has a length of 120 units and traverses arcs $(1,2),(2,9),(9,14),(14,16)$.

Notice that the targeted arc for the notional example has the smallest interdiction cost along the shortest path. However, the arc selected for targeting is not necessarily the minimum cost arc on the shortest path. Moreover, the model is limited in that it does not indicate the increase in shortest path length in the residual network. Further, notice that the objective function value does not calculate the interdiction cost associated with targeting the node indicated by the z-variable.

These drawbacks may render the single shortest path problem less desirable, but they are required to allow the interdiction of more than one shortest path. A problem closely related to the SAD model is to determine the k arcs that should be targeted so as to interdict the k shortest arc independent paths from the source to the terminus. If the number of arcs contained in the cut set is less than or equal to k, the minimum cost cut set interdicts all shortest paths between the source and terminus. The decision variables are as described for the SAD model. The x-variable is indexed to account for the k multiple paths: x_{ij}^{ℓ}

takes value 1 if arc (i, j) is on the ℓth shortest arc-independent path from the source to the terminus and has a value of 0, otherwise.

In this model, only *arc-independent* paths are considered. In other words, the k shortest paths will consider paths that do not contain any arcs in common. To ensure the paths determined in the model are arc-independent, an arc that is contained in a shortest path may not be utilized for another path. Therefore, the x-variable must by constrained as follows

$$x_{ij}^{\ell} \leq 1 - \sum_{h=1}^{\ell-1} x_{ij}^{h}, \forall (i, j) \in E, \forall \ell = 2, \ldots, k.$$

The formulation for the k-shortest arc-independent paths arc destroying problem (**SAD-kI**) follows:

$$\min \quad \sum_{(i,j)\in E} c_{ij}(y_{ij}+z_{ij}) + \lambda \sum_{\ell=1}^{k} \sum_{(i,j)\in E} d_{ij} x_{ij}^{\ell} \tag{4.3a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} + z_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{4.3b}$$

$$u_s = 0, \tag{4.3c}$$

$$u_t = 1, \tag{4.3d}$$

$$\sum_{j:(i,j)\in E} x_{ij}^{\ell} - \sum_{j:(j,i)\in E} x_{ji}^{\ell} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V, \forall \ell = 1,\ldots,k, \tag{4.3e}$$

$$x_{ij}^{\ell} \leq 1 - y_{ij}, \qquad \forall (i,j) \in E, \forall \ell = 1,\ldots,k, \tag{4.3f}$$

$$\sum_{(i,j)\in E} z_{ij} = k, \tag{4.3g}$$

$$x_{ij}^{\ell} \leq 1 - \sum_{h=1}^{\ell-1} x_{ij}^{h} \qquad \forall (i,j) \in E, \forall \ell = 2,\ldots,k, \tag{4.3h}$$

$$x_{ij}^{\ell} \in \{0,1\}, \qquad \forall (i,j) \in E, \forall \ell = 1,\ldots,k, \tag{4.3i}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.3j}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.3k}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{4.3l}$$

The SAD-kI model identifies the minimum cost cut set and further identifies the k arcs in the cut set that interdict the k arc-independent shortest paths. The decision variable z takes value 1 for the 1st through kth shortest paths.

Constraint (4.3b) ensures a minimum cost cut set and selects the arcs in the set along each of the k arc-independent shortest paths; *i.e.*, the k shortest paths do not contain any

arcs in common. It is possible that multiple paths traverse the same node, but the constraint ensures that an arc is utilized only once in a shortest path. Constraint (4.3h) ensures that arcs utilized for shorter s-t paths are not traversed for subsequent shortest paths. All remaining constraints are as described for the SAD model.

In the context of the notional military transportation network example given in Figure 14 and Table 42, the $k = 3$ shortest paths are determined using the SAD-kI model. To interdict the k shortest, arc independent paths, arcs {(2,9), (3,6), (5,12)} should be targeted for destruction at a cost of 11 units. The minimum cut solution (indicated via the y- and z-variables) is to interdict arcs {(2,9), (2,6), (3,6), (7,10), (11,14), (11,15), (8,12), (5,12)} at a total interdiction cost of 34. The shortest path utilizing the capacities given in the table as distances is 120 units in length, which is the solution for the SAD model, and traverses arcs $(1, 2), (2, 9), (9, 14), (14, 16)$. The remaining two shortest paths respectively traverse arcs $(1, 5), (5, 12), (12, 15), (15, 16)$ and $(1, 3), (3, 6), (6, 9), (9, 13), (13, 16)$ with costs of 180 and 190 units. The model is infeasible for $k = 4$ indicating that there are no more than three arc-independent s-t paths. In this network instance, although there are three destination nodes for the supplies, the artificially installed arcs from each of nodes 13, 14, and 15 to node 16 are included in the limitation of arc independent paths, so there are no additional paths for the model to select. This feature of the model, while a drawback for this particular instance, may be a benefit for discovering bottlenecks in other networks of interest.

The drawback of utilizing arc-independent paths is illustrated using a notional network. Consider the network depicted in Figure 16. The arc interdiction costs are indicated above each arc in the network. Assume the length of each arc is 1 unit. The minimum cut set solution for the SAD-kI model with $k = 1$ is to interdict arcs (1,3) and (2,3) at a cost of 2 units. Either arc in the cut set is along the shortest path, so the model may arbitrarily select arc (1,3) to be the targeted arc. The solution when $k = 2$ is infeasible since arc $(3, t)$

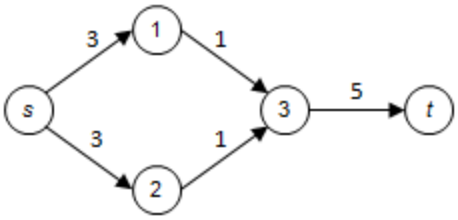


Figure 16. Notional network example with arc-independence

is already along a path that is interdicted. This example illustrates the necessity to consider an alternative approach.

The SAD-kI model is extended to relax the constraint of arc-independence. The decision variable z also takes an additional index so it is related to the k shortest paths. Let z_{ij}^{ℓ} indicate whether arc (i, j) is on the ℓth shortest path and is included in the cut set; *i.e.*, arc (i, j) is targeted to interdict the ℓth shortest path. The k shortest path destroy model that

does not utilize arc independence (**SAD-k**) follows:

$$\min \quad \sum_{(i,j)\in E} c_{ij}\left(y_{ij} + \sum_{\ell=1}^{k} z_{ij}^{\ell}\right) + \lambda \sum_{\ell=1}^{k} \sum_{(i,j)\in E} d_{ij} x_{ij}^{\ell} \tag{4.4a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} + \sum_{\ell=1}^{k} z_{ij}^{\ell} \geq 0, \qquad \forall (i,j) \in E, \tag{4.4b}$$

$$u_s = 0, \tag{4.4c}$$

$$u_t = 1, \tag{4.4d}$$

$$\sum_{j:(i,j)\in E} x_{ij}^{\ell} - \sum_{j:(j,i)\in E} x_{ji}^{\ell} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V, \forall \ell = 1, \ldots, k, \tag{4.4e}$$

$$x_{ij}^{\ell} \leq 1 - y_{ij}, \qquad \forall (i,j) \in E, \forall \ell = 1, \ldots, k, \tag{4.4f}$$

$$\sum_{\ell=1}^{k} \sum_{(i,j)\in E} z_{ij}^{\ell} = k, \tag{4.4g}$$

$$\sum_{(i,j)\in E} z_{ij}^{\ell} = 1, \qquad \forall \ell = 1, \ldots, k, \tag{4.4h}$$

$$x_{ij}^{\ell} \leq 1 - \sum_{h \neq \ell} z_{ij}^{h} \qquad \forall (i,j) \in E, \forall \ell = 1, \ldots, k, \tag{4.4i}$$

$$x_{ij}^{\ell} \in \{0,1\}, \qquad \forall (i,j) \in E, \forall \ell = 1, \ldots, k, \tag{4.4j}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.4k}$$

$$z_{ij}^{\ell} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{4.4l}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{4.4m}$$

The SAD-k model identifies the minimum cost cut set and further identifies the k arcs in the cut set that interdict the k shortest paths, taking into account the arc interdicted for

the shortest path. Note that the ℓ index does not force the k shortest paths to be ordered by the index. Additional constraints are required to order the shortest paths.

Constraints (4.4b) and (4.4f) ensure a minimum cost cut set is selected containing arcs along each the k shortest paths; *i.e.*, the k shortest paths do not contain arcs within the cut in common. For each of the k shortest paths, a single arc on the path is allowed in the cut set via Constraints (4.4f) and (4.4h). All remaining constraints are as described for the SAD-kI model.

The example network depicted in Figure 14 and Table 42 representing a notional military transportation network is solved with the SAD-k model with $k = 3$. The arcs to target are {(2,9), (3,6), (5,12)} at a total cost of 11 units. The k shortest paths have respective lengths of 120, 160, and 160 units and traverse arcs $(1,2),(2,9),(9,14),(14,16)$, $(1,3),(3,6),(6,9),(9,14),(14,16)$, and $(1,5),(5,12),(12,14),(14,16)$, respectively. Note that the shortest paths are not path independent and have arcs in common. The model ensures that the shortest paths traverse the cut set via independent arcs. The cut set (indicated via the y and z variables) contains arcs {(2,9), (2,6), (3,6), (7,10), (11,14), (11,15), (8,12), (5,12)} which would cost 34 units to interdict. In contrast to the SAD-kI model, the SAD-k model is feasible for $k = 4$ as there is an additional shortest path through the cut set. In fact, if the value of k is larger than the size of the minimum cost cut set, a larger cut set is allowed so as to identify the appropriate number of shortest paths. There is, of course, a k-value at which the model will be infeasible; however, the model will have to enumerate all shortest paths to identify it, which would be inefficient.

The leader's portion of the arc interdiction models in this chapter can be extended as described by Kennedy *et al.* [45] to target nodes rather than arcs. Additionally, Kennedy *et al.* demonstrate extending the leader's cut set model to target both nodes and arcs [45].

4.4 Summary

This chapter developed a number of models for destroying arcs in a network. The contributions are summarized in Table 45. Models representing arc interdiction were developed to determine the arcs that should be targeted to interdict the k shortest paths. In the next chapter, network diverting tasks are considered.

Table 45. Destroying Interdiction Task Contributions

Interdiction Task	Section	Contribution	Description
Destroy	4.2	MAD	The maximum flow arc destroying model ensures that the leader's minimum cost source-terminus cut set is selected eliminating the follower's flow.
	4.3	SAD	The shortest path arc destroying model identifies a single arc in the intersection of the leader's source-terminus cut set and the follower's shortest path.
	4.3	SAD-kI	Extends SAD model. The model selects k arcs in the leader's minimum cost cut set to interdict the k shortest arc-independent paths.
	4.3	SAD-k	Extends SAD model. The model identifies the k arcs in the leader's minimum cut set that interdict the follower's k shortest paths.

V. Divert Interdiction Tasks

5.1 Introduction

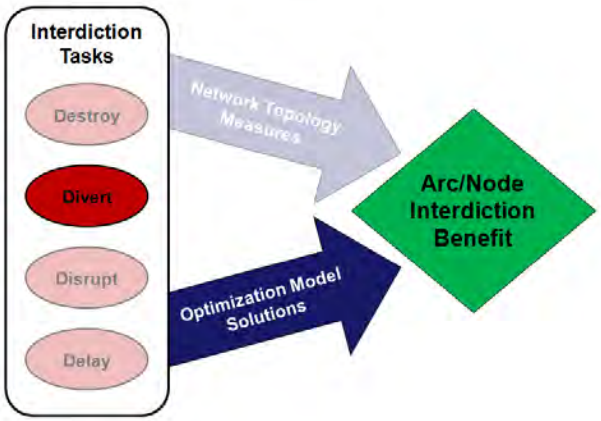


Figure 17. Research Framework: Modeling Divert Interdiction Tasks

In this chapter, a network interdiction modeling framework is developed for diverting network resources, flow, or traffic away from (*i.e.*, not to be routed to or through) a predefined set of nodes. This chapter addresses the 'divert' interdiction task within the models approach of the research framework as depicted in Figure 17.

In the open literature, a great deal of research has examined the network diversion problem, which is summarized in Section 2.3.2 of the literature review. However, these models actually align with the *channeling* objective for interdiction tasks within joint doctrine. Channeling is forcing the enemy to move forces, supplies, or communications along specific, more exploitable routes [1:p. I-8].

This chapter is concerned with solving the *network diverting problem* (NDP). Given a network topology and a designated source-terminus pair, an actor (leader) seeks to divert a path or flow from a set of nodes, termed the *divert set*, by affecting the network's arcs and/or nodes. Subsequently, a network operator (follower) seeks to optimize the path or

flow of the residual network. The divert set may be used to model targeting preferences in military campaign planning. Conversely, the divert set can model critical assets that must be defended. The NDP is solved over a single-layered network while utilizing the destruction of nodes and/or arcs.

The NDP is illustrated in the context of a convoy commander's decision process. The road network through which the convoy travels contains areas occupied by adversaries with various levels of forces and capabilities. The commander seeks to route the convoy safely to its destination by diverting the route from areas that have the largest risk of enemy action. In this scenario, the divert set represents the regions through which travel is too risky, and the convoy commander can be considered to be both the leader and follower, diverting from hostilities and routing the convoy along the shortest path with acceptable risk.

Alternatively, consider a military conflict within a city. The city's roads and bridges form the arcs of the network and the road intersections are the nodes. The leader is an insurgent force, and the follower is the military's security element patrolling the city's streets. The insurgent forces block roads or bridges to force the security patrols onto alternate routes and away from the intersection where a guerrilla raid is planned against a third party. The security element seeks a maximum flow of patrols through the city's streets, which are partially blocked by the insurgents. The result of the interdiction actions is that military patrols are diverted from the intersection of the insurgent raid.

A non-military illustration of the network diverting problem involves a city that will host a festival on a portion of its downtown streets. Due to the large volume of pedestrians, vehicle traffic should be diverted from the area. The city manager's office must select the roads and bridges to close for non-pedestrian traffic to divert all vehicle traffic from the festival. The roads and bridges form the network (intersections are nodes), the city manager is the leader, and the traffic is the follower. The divert set consists of the intersections where

the festival will take place. The maximum flow on the network represents traffic flow, and the city manager desires to eliminate flow through the divert set.

Another application of the network diverting problem involves determining detour routes around closed roads. Alternatively, a supply chain may seek to avoid certain countries or routes to avoid tariffs, taxes or tolls. In addition, when it is determined that a portion of a communications network has been hacked or compromised, it is vital that important or sensitive information be routed around such sections of the network. Finally, the freight industry seeks to avoid large metropolitan areas during peak traffic times to reduce the risk of being delayed by the congestion or to avoid toll roads or bridges.

The network diverting problem requires that both the leader and the follower be included in any models that are generated. It is assumed that the leader must allow the follower to divert, *i.e.*, a shortest path or maximum flow for the follower will exist in the residual network. Therefore, the follower's problem is a consideration for the leader when making interdiction decisions. Likewise, the follower optimizes the path or flow on the residual network only after the leader has made a decision. For these reasons, both the leader and follower are included in the model to solve the network diverting problem.

A bilevel programming problem is the mathematical formulation of a non-cooperative sequential game between two players (termed leader and follower) who take turns in a one-move game in which each player is assumed to have all information about the problem. In economic literature, this is a Stackelberg game. Let x and y denote the decision variables of the leader and follower, respectively. The objective functions of the leader and follower are denoted $F(x, y)$ and $f(x, y)$, respectively. The constraint functions of the leader and follower are respectively denoted $G(x, y)$ and $g(x, y)$. The general mathematical formulation [5:p. 6]

is

$$\begin{aligned} \min_{x \in X} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0, \\ & \min_{y \in Y} \quad f(x, y), \\ & \text{s.t.} \quad g(x, y) \leq 0. \end{aligned}$$

Note that there is an optimization problem with the constraints. The second optimization problem, $\min_{y \in Y} f(x, y)$, is typically called the lower-level, inner, or follower's problem. The context of the NDP fits nicely into this formulation.

First, consider the leader's problem. It is assumed that the network has no negative cycles, the divert set contains only nodes, and the attacker targets only arcs. The objective of the attacker is to eliminate flow to the divert set. The overall flow from the source to the terminus is not the main consideration for the leader. One method to eliminate flow in a network is to identify a cut set. In this case the attacker seeks to cut the source s from the divert set D. Alternatively, the attacker may cut the divert set from the terminus t. The resulting s-D or D-t cut accomplishes the goal of diverting flow from D. To ensure diverting, the leader will not select an interdiction strategy that results in an s-t network cut.

The choice of making the cut on the source or terminus side of D is arbitrary but may not be equivalent; there may be cost or non-quantifiable reasons for selecting one over the other. As such, it may be advisable to solve both problems and select the one with the smallest cost. Unless noted otherwise, a s-D cut is used in the remainder of this work. The same methodology applies for D-t cuts.

Within this research, a cut set is utilized to separate the divert set from the rest of the network. This task may be completed by utilizing the isolation set problem. The leader's problem would utilize an isolation model similar to Bellmore *et al.* [7] as well as Herbranson

et al. [40]. While this approach is not utilized in this research, it may be a fruitful avenue for future research.

Consider the follower's problem. The follower employs the standard shortest path or maximum flow formulation subject to the needs of a particular instance or scenario. The leader attempts to destroy arcs and/or nodes to halt flow to the divert set if the follower's objective is maximizing flow or to disallow a shortest path to traverse the divert set if the follower is minimizing shortest path length. The follower must react to the decisions of the leader, so the leader's decisions influence the corresponding formulation.

In the bilevel programming formulation of the network diverting problem, the leader's minimum cut problem takes the place of F (objective) and G (constraints). The appropriate shortest path or maximum flow formulation is used for the follower, with the follower's problem taking the place of f (objective) and g (constraints).

Bracken and McGill [9] discuss the fact the leader decides first and then the follower, given that both are assumed to have complete knowledge of the other's objective and constraints. This is valid because, regardless of the choice the follower makes, the leader's optimization problem must be feasible for the follower's problem. Thus, whether the leader chooses first or the leader and follower choose simultaneously, the solution is the same. Because the decision variables for the leader and follower in network diverting models can be determined simultaneously, a single-objective multi-criteria mathematical program is utilized rather than a bilevel programming formulation.

The remainder of the chapter proceeds as follows. The network diverting problem is formulated for maximum flow networks. Next, shortest path models are considered for network diverting. In each of these sections, testing of several network structures is conducted to demonstrate the models' utility. Then, model extensions are presented for both flow and path models. Finally, the relationship between diverting and channeling is explored for directed networks.

5.2 Maximum Flow Diverting

The NDP is formulated by partitioning the model into the leader's problem and the follower's problem. Recall that D denotes the divert set. The leader's problem is formulated as the typical minimum cut [6:p.598] and is similar to the presentation of Wood [68]. The leader's decision variable y_{ij} indicates whether arc (i, j) is interdicted ($y_{ij} = 1$) or not ($y_{ij} = 0$). The variable u_i denotes whether node i is on the source-side of the s-D cut ($u_i = 0$) or the D-side ($u_i = 1$). The follower's problem is formulated using the typical maximum flow model that adds an artificial, unbounded capacity return arc from the destination to the source [6:p. 596]. The follower's decision variable x_{ts} indicates the flow on the return arc (t, s). The return arc is added to the model from the terminus to the source and flow is maximized due to the flow conservation constraints. The augmented set of arcs in the network is $E' = E \cup (t, s)$.

A constraint is included in the maximum flow NDP model to ensure that the leader allows at least some flow for the follower. Without this constraint the leader's interdiction strategy might not result in diverting.

The follower's objective is combined with the leader's by subtracting it, thus preserving its maximization:

$$\sum_{(i,j)\in A} c_{ij} y_{ij} - \lambda x_{ts}$$

(leader) (follower)

The magnitude of the weight of the follower's objective is given by λ and can be considered a cooperation weight. Typically, the leader has no concern for the follower's objective value (non-cooperative assumption). However, in the festival example, the city manager may

choose to allow less restrictive traffic patterns by changing the value of λ. Increasing λ gives a follower more relative influence and is equivalent to a relaxing assumption.

Under non-cooperative conditions, care must be taken to ensure that the leader's objective is preferred over the follower's, *i.e.*, the leader decides first. This is accomplished by assigning the weight of the follower's objective small enough that, no matter the costs or capacities, the leader's objective dominates the objective. Moreover, the value of λ determines which methodology the solver will utilize. When the value is very small, the leader's minimum cut objective will be the primary feature the solver optimizes. An added benefit is that minimum cut models typically solve quickly.

The formulation for the maximum flow network diverting problem for arcs only (**MNDP-a**) follows:

$$\min \quad \sum_{(i,j)\in E} c_{ij}y_{ij} - \lambda x_{ts} \tag{5.1a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.1b}$$

$$u_s = 0, \tag{5.1c}$$

$$u_k = 1, \qquad \forall k \in D \subset V, \tag{5.1d}$$

$$\sum_{j:(i,j)\in E'} x_{ij} - \sum_{j:(j,i)\in E'} x_{ji} = 0, \qquad \forall i \in V, \tag{5.1e}$$

$$x_{ij} \leq b_{ij}(1 - y_{ij}), \qquad \forall (i,j) \in E, \tag{5.1f}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.1g}$$

$$x_{ts} \geq 1, \tag{5.1h}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.1i}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{5.1j}$$

The objective (5.1a) ensures that the leader's minimum cost s-D cut set is selected while maximizing the remaining source-to-terminus flow on the network. The cost to interdict arc (i, j) is represented by c_{ij}. To ensure that the leader has preemptive preference over the follower, the weight for λ can be determined by utilizing Sherali's Algorithm 1 [58].

The leader's problem consists of the first term of the objective function and constraints (5.1b)-(5.1d) and (5.1h)-(5.1j). Constraint (5.1b) ensures that a cut is employed. The source nodes are identified in (5.1c). In Constraint (5.1d) the divert set D is treated as a destination in the minimum cut formulation and the nodes in the divert set are assigned $u_k = 1$ to enforce the cut after the source but before (and outside) the divert set. Notice that the nodes in the divert set can not be interdicted even if nodal interdiction is allowed. Finally, Constraints (5.1i) and (5.1j) ensure the leader's decision variable values are binary.

The follower's problem is the maximum flow problem and consists of the last term in the objective function and constraints (5.1e)-(5.1g). Constraint (5.1e) ensures flow conservation at each node. The flow on each arc is non-negative (5.1g). Constraint (5.1f) ensures the flow is zero if the leader interdicts the arc, otherwise it is bounded above by the arc's capacity b_{ij}. Constraint (5.1h) enforces the assumption that a flow will exist for the follower in the residual network.

To modify the formulation to find a D-t cut, the only change is to Constraints (5.1c) and (5.1d). To enforce such a cut, these are changed to $u_t = 1$ and $u_k = 0, \forall k \in D \subset N$, thus forcing a cut between the divert set and the terminus.

In practice, it is only necessary that the y_{ij}- or u_i-variables be binary to ensure the leader's decision takes a value of 0 or 1 [19]. Therefore, since there are fewer nodes than arcs, the u_i-variables are restricted to be binary-valued and the y_{ij}-variables are continuous. In addition, provided the arc capacities are integer-valued, the x_{ij}-variables will take on integer values and are continuous in the model.

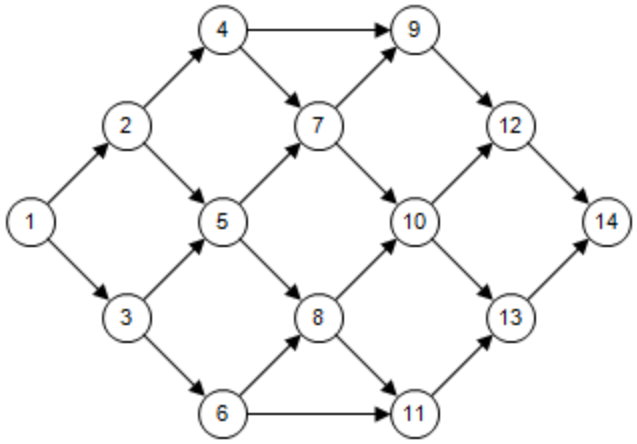


Figure 18. Example Network

A simple illustration demonstrates that a leader must be open to considering either *s*-*D* or *D*-*t* cuts. Consider the network in Figure 18 with a divert set consisting of nodes 11 and 13. The interdiction cost of every arc is one unit. The minimum *s*-*D* cut (that also allows a diverting flow for the follower) requires three arc cuts (arcs {(6,11), (8,11), (10,13)}) to eliminate flow to the divert set. On the other hand, the minimum *D*-*t* cut would require only one arc cut (arc (13,14)). Both solutions ensure there is no flow to the divert set via the flow conservation constraints, but unless there are non-quantified reasons such as the decision maker's preference, the leader should select the *D*-*t* cut formulation and solution as it results in a lower cost.

A bound for the leader's problem can be determined by examining the divert set. A worst-case bound for the interdiction cost is the sum of the interdiction costs of all arcs entering (or leaving) the divert set. A lower cost interdiction strategy may be available, but it can be no larger since the leader will select arcs to interdict so as to cut flow into (or equivalently out of) the divert set, driving the total interdiction cost down as sought by the objective. The choice of the inbound or outbound arcs of the divert set should align with the cut chosen in the model. If the *s*-*D* cut is modeled, inbound arc costs should be summed.

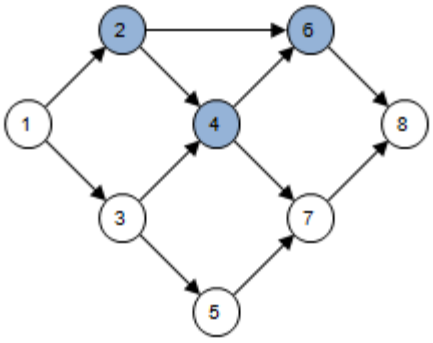


Figure 19. A notional directed network and possible divert set

Conversely, if a D-t cut is modeled, a cut consisting of the outbound arcs must be used to determine the bound.

5.2.1 Numerical Examples.

Numerical examples illustrate the MNDP-a models. Consider the directed network depicted in Figure 19 with shaded divert set $D = \{2, 4, 6\}$. Each arc has a unit interdiction cost and capacity. The optimal solution of the MNDP-a model is a cut cost to the leader of 2 units with the follower's maximum flow being 1. The cut consists of arcs (1,2) and (3,4). For this notional network topology, there exists an equivalent minimum cost solution which eliminates all network flow, but it does not allow diverting and may not be selected.

Next, consider a military battle being fought on three fronts and an adversary using four supply depots from which to deliver military items. Figure 20 depicts this notional military transportation network example as used by Ghare *et al.* [33] and Wood [68]. The capacity and interdiction costs are enumerated in Table 46. Suppose that reports indicate that the battle will change so that the main fronts will now be at nodes 9 and 10. Given this new information, the divert set would be the new battle fronts and the goal is to ensure the adversary cannot provide supplies to the new battle space. When this problem is solved

Arc	Capacity	Cost	Arc	Capacity	Cost
(1,2)	1000	100	(6,10)	60	7
(1,3)	1000	100	(7,10)	120	4
(1,4)	1000	100	(7,11)	150	6
(1,5)	1000	100	(8,11)	120	6
(2,6)	60	5	(8,12)	80	4
(2,9)	70	4	(9,13)	80	4
(2,7)	60	5	(9,14)	50	5
(3,6)	50	3	(10,13)	100	5
(3,7)	50	3	(10,14)	80	4
(3,8)	60	5	(11,14)	180	6
(4,7)	100	3	(11,15)	100	4
(4,8)	80	5	(12,14)	80	5
(5,7)	50	5	(12,15)	100	6
(5,8)	100	5	(13,16)	1000	100
(5,12)	80	4	(14,16)	1000	100
(6,9)	60	4	(15,16)	1000	100

Table 46. Data for the notional military transportation network in Figure 20 [33, 68]

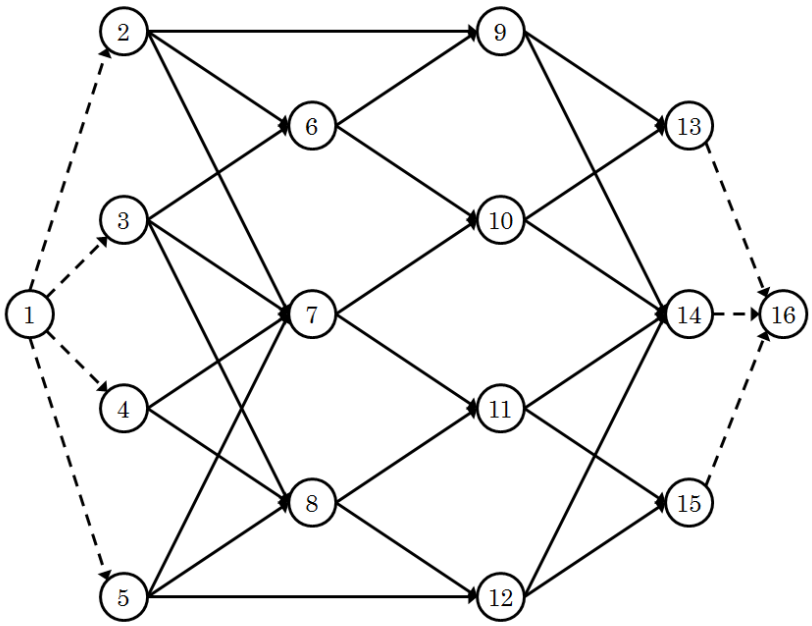


Figure 20. A notional military transportation network [33, 68]

using the MNDP-a model, the solution is to interdict arcs {(2,9), (2,6), (3,6), (7,10)} at a total interdiction cost of 16. The maximum flow for the residual network is 430 units.

5.2.2 Testing and Results.

The MNDP-a model is tested on network instances that are generated as described in Appendix A. The parameters for each network structure utilized are given in Table 47. For each of the 24 network structure cases listed, 10 replicates were generated. For each replicate, 10 unique divert sets were constructed using the following procedure.

An arbitrary node not connected to the source or terminus is placed in the set J. Subsequently, a node that is connected to the set J (and not connected to the source or terminus) is added to the set. This is repeated until there are between 5 and 10 nodes in the set. In

Table 47. Parameter settings for 100-node random network structures

Structure	Parameters	Case	Mean Density
ER	$\beta = 0, p = 0.1$	1	0.098
	$\beta = 1, p = 0.1$	2	0.099
	$\beta = 0, p = 0.5$	3	0.5
	$\beta = 1, p = 0.5$	4	0.502
BA	$m_a = 2, n_0 = 25$	5	0.091
	$m_a = 5, n_0 = 25$	6	0.136
	$m_a = 2, n_0 = 55$	7	0.318
	$m_a = 10, n_0 = 50$	8	0.348
WS	$k = 4, p = 0.10$	9	0.04
	$k = 4, p = 0.25$	10	0.04
	$k = 30, p = 0.10$	11	0.303
	$k = 30, p = 0.25$	12	0.303
PNDCG	$\beta = 0, \alpha = 2.35$	13	0.025
	$\beta = 1, \alpha = 2.35$	14	0.025
	$\beta = 0, \text{dist} = U_{3,5}$	15	0.03
	$\beta = 1, \text{dist} = U_{3,5}$	16	0.03
	$\beta = 0, \text{dist} = U_{20,5}$	17	0.202
	$\beta = 1, \text{dist} = U_{20,5}$	18	0.202
Grid	10×10	19	0.037
	5×20	20	0.038
	20×5	21	0.035
Star	10×10	22	0.04
	5×20	23	0.04
	20×5	24	0.04

some instances, the size of the set is allowed to contain fewer nodes because there exist no other nodes connected to the set.

The values of the interdiction cost-weight and the capacity-weight assigned to each arc in the network are selected using a discrete (integer) uniform distribution having a range between 1 and 10, inclusively.

The MNDP-a model is solved for the various random network instances using IBM® ILOG® CPLEX® Optimization Studio V12.6. For each instance, two solutions are determined. First, the MNDP-a model is solved with no initial solution provided. Second, the MNDP-a model is solved using the worst-case bound of cutting all inbound arcs to the divert set for s-D model. Alternatively, the MNDP-a model is altered and solved using a D-t cut and the outbound arcs from the divert set are utilized as the alternate initial solution.

The solutions when the MNDP-a model (s-D cut) is solved both without an initial solution and utilizing the initial solution of the worst-case cut (all arcs into the divert set) are compared in Table 48. The solutions when the MNDP-a model (D-t cut) is solved having no initial solution and utilizing the worst-case cut (all arcs out of the divert set) as the initial solution are compared in Table 49. Finally, the s-D and D-t cut models are compared to determine whether there is a difference in the solution values or computation times in Table 50. For each of the solution approaches, the number of instances solved to optimality, mean solution time, and solution time standard deviation are reported. Moreover, the number of network instances that resulted in unequal solution values is tabulated. The average difference, if one exists, is also included.

For each of the comparisons, a two-tailed paired-t test statistic is utilized to determine whether there is a difference in the solution times when comparing the two alternatives. In cases when the p-value is less than 0.05, there is sufficient evidence at the 0.05 level of significance to reject the claim that there is no difference in the solution times. For larger

p-values, there is insufficient evidence at the 0.05 level of significance to reject the claim of no difference in the solution times.

Examination of the p-values comparing mean solution times for either s-D or D-t cuts with and without initial solutions in Tables 48 and 49, respectively, show values larger than 0.05 for nearly all network cases tested. Within the tests cases with smaller p-values, three cases suggest shorter solution times when utilizing no initial solution, whereas four cases suggest that providing an initial worst-case solution leads to faster solutions. Therefore, there is insufficient evidence at the 0.05 level of significance to reject the claim of no difference in the solution times in the cases tested. The decision to use initial solutions related to either inbound or outbound arc cuts versus providing no initial solution is arbitrary concerning solution times for the network instances tested.

The p-values when comparing the solution times of the s-D and D-t cut models suggest that, for the network instances tested, there may be benefit to selecting one over the other for a given network structure. For example, for the BA network structure cases tested, there is sufficient evidence at the 0.05 level of significance to reject the claim of no difference in the solution times. Thus, these results suggest that the s-D cut model is preferred over the D-t cut model for the BA networks tested. The construction of the BA network structure informs this tendency. The high density of arcs close to the source may allow more rapid identification of cuts for the s-D model, whereas the sparseness of the network outside the initial set of nodes in the construction of these networks may prevent the model from quickly selecting arcs for the cut in the D-t model.

Furthermore, for the network instances tested, the D-t model yielded an optimal solution in every case; the s-D model did not yield optimal solutions for every test. A general conclusion is difficult to provide. Perhaps additional testing can clarify whether there exists skewing in the random selection of nodes included in the divert sets tested. However, this result does reveal the importance of decision makers being open to solutions from either

model. Additionally, it demonstrates that analysts should examine all possible models before making recommendations to a decision maker if it is practical to do so.

The grid network cases tested highlight this observation. There is a solution value decrease in each of the network cases tested (*i.e.*, 10×10, 5×20, and 20×5 grid structures). That is, the *D-t* solution values require less costs as indicated by the average change value being larger than 1. A change near the value of λ suggests that the arc cut required the same interdiction cost but yielded a solution that utilized a shortest path length different than the *s-D* model solution. For the tested network instances, the *D-t* cuts identified arc cuts that are, on average, 3 cost units less expensive. In all other network structure cases tested, four test instances observed a difference in solution values. The difference for these instances was the result of a path length change, not an interdiction cost change.

Table 48. Comparison of results for s-D cuts with and without providing initial solution

Structure	Case	No initial solution			Cut Inbound Arcs			Solutions		2-Tail
		# Solved	Mean Time	StDev	# Solved	Mean Time	StDev	# ≠	Mean Change	p-value
ER	1	100 / 100	33.384	77.795	100 / 100	33.370	77.072	0	-	0.977
	2	100 / 100	32.805	57.969	100 / 100	31.537	56.182	0	-	0.043
	3	100 / 100	138.151	157.620	100 / 100	150.416	172.588	0	-	0.033
	4	100 / 100	150.000	105.719	100 / 100	156.355	108.856	0	-	0.034
BA	5	100 / 100	5.818	50.742	100 / 100	6.479	56.932	0	-	0.291
	6	100 / 100	10.990	43.682	100 / 100	11.728	48.044	0	-	0.148
	7	100 / 100	4.185	16.401	100 / 100	3.640	12.924	0	-	0.464
	8	99 / 100	31.821	107.335	99 / 100	33.571	125.797	0	-	0.607
WS	9	100 / 100	2.936	15.397	100 / 100	3.037	16.974	0	-	0.546
	10	100 / 100	3.987	11.121	100 / 100	4.175	12.268	0	-	0.388
	11	100 / 100	176.194	279.721	100 / 100	178.632	275.665	0	-	0.564
	12	100 / 100	83.765	72.631	100 / 100	81.256	63.365	0	-	0.440
PNDCG	13	89 / 100	37.837	190.769	89 / 100	43.064	221.989	0	-	0.488
	14	91 / 100	1.108	4.079	91 / 100	1.300	5.492	0	-	0.437
	15	98 / 100	1.074	1.804	98 / 100	1.090	1.896	0	-	0.834
	16	99 / 100	1.818	10.804	99 / 100	1.604	9.129	0	-	0.282
	17	100 / 100	35.296	33.044	100 / 100	35.676	33.791	0	-	0.683
	18	100 / 100	46.639	109.968	100 / 100	48.368	112.648	0	-	0.087
Grid	19	100 / 100	2.058	4.398	100 / 100	2.048	4.200	0	-	0.895
	20	100 / 100	0.043	0.015	100 / 100	0.027	0.009	0	-	< 0.001
	21	94 / 100	10.023	23.107	94 / 100	9.174	15.428	0	-	0.439
Star	22	100 / 100	38.584	150.159	100 / 100	38.660	153.248	0	-	0.909
	23	100 / 100	0.645	4.805	100 / 100	0.600	4.506	0	-	0.134
	24	79 / 100	155.238	503.811	78 / 100	148.610	491.684	0	-	0.943

Table 49. Comparison of results for D-t cuts with and without providing initial solution

Structure	Case	No initial solution			Cut Outbound Arcs			Solutions		2-Tail
		# Solved	Mean Time	StDev	# Solved	Mean Time	StDev	# ≠	Mean Change	p-value
ER	1	100 / 100	17.172	18.839	100 / 100	16.519	18.087	0	-	0.115
	2	100 / 100	40.825	51.815	100 / 100	37.502	46.915	0	-	0.071
	3	100 / 100	166.135	172.571	100 / 100	179.745	194.659	0	-	0.025
	4	100 / 100	284.425	241.893	100 / 100	270.274	224.040	0	-	0.086
BA	5	99 / 100	35.336	206.807	100 / 100	64.163	356.172	0	-	0.719
	6	96 / 100	78.856	194.284	97 / 100	114.066	381.318	0	-	0.476
	7	100 / 100	4.491	18.275	100 / 100	4.873	18.559	0	-	0.597
	8	98 / 100	108.621	346.368	97 / 100	80.622	157.335	0	-	0.392
WS	9	100 / 100	0.891	1.192	100 / 100	0.907	1.312	0	-	0.738
	10	100 / 100	7.519	16.288	100 / 100	6.573	13.562	0	-	0.061
	11	100 / 100	182.233	259.743	100 / 100	187.349	262.850	0	-	0.177
	12	100 / 100	106.733	89.597	100 / 100	105.328	87.058	0	-	0.582
PNDCG	13	89 / 100	21.190	170.152	89 / 100	26.641	212.422	0	-	0.228
	14	91 / 100	0.149	0.396	91 / 100	0.172	0.603	0	-	0.467
	15	98 / 100	3.020	10.844	98 / 100	3.127	12.239	0	-	0.570
	16	99 / 100	0.840	1.873	99 / 100	0.894	2.141	0	-	0.347
	17	100 / 100	28.965	22.885	100 / 100	28.222	21.708	0	-	0.275
	18	100 / 100	50.390	120.592	100 / 100	51.119	125.417	0	-	0.827
Grid	19	100 / 100	2.722	5.295	100 / 100	2.616	5.143	0	-	0.380
	20	100 / 100	0.032	0.013	100 / 100	0.031	0.014	0	-	0.854
	21	94 / 100	5.867	10.676	94 / 100	6.140	13.226	0	-	0.496
Star	22	94 / 100	196.849	526.998	93 / 100	160.325	413.676	0	-	0.275
	23	100 / 100	16.689	106.388	100 / 100	17.079	112.068	0	-	0.516
	24	79 / 100	107.067	252.888	78 / 100	99.156	276.360	0	-	0.899

Table 50. Comparison of results for s-D and D-t cuts

Structure	Case	s-D Cut			D-t Cut			Solutions		2-Tail
		# Solved	Mean Time	StDev	# Solved	Mean Time	StDev	# ≠	Mean Change	p-value
ER	1	100 / 100	33.384	77.795	100 / 100	17.172	18.839	0	-	0.015
	2	100 / 100	32.805	57.969	100 / 100	40.825	51.815	0	-	0.124
	3	100 / 100	138.151	157.620	100 / 100	166.135	172.571	0	-	0.002
	4	100 / 100	150.00	105.719	100 / 100	284.425	241.893	0	-	< 0.001
BA	5	100 / 100	5.818	50.742	100 / 100	35.336	206.807	0	-	0.097
	6	100 / 100	10.990	43.682	100 / 100	78.856	194.284	0	-	< 0.001
	7	100 / 100	4.185	16.401	100 / 100	4.491	18.275	0	-	0.86
	8	99 / 100	31.821	107.335	100 / 100	108.621	346.368	0	-	0.011
WS	9	100 / 100	2.936	15.397	100 / 100	0.891	1.192	0	-	0.169
	10	100 / 100	3.987	11.121	100 / 100	7.519	16.288	1	0.00013	0.018
	11	100 / 100	176.194	279.721	100 / 100	182.233	259.743	0	-	0.386
	12	100 / 100	83.765	72.631	100 / 100	106.733	89.597	0	-	0.001
PNDCG	13	89 / 100	37.837	190.769	100 / 100	21.190	170.152	1	0.00001	0.545
	14	91 / 100	1.108	4.079	100 / 100	0.149	0.396	0	-	0.028
	15	98 / 100	1.074	1.804	100 / 100	3.020	10.844	0	-	0.062
	16	99 / 100	1.818	10.804	100 / 100	0.840	1.873	0	-	0.371
	17	100 / 100	35.296	33.044	100 / 100	28.965	22.885	0	-	0.023
	18	100 / 100	46.639	109.968	100 / 100	50.390	120.592	0	-	0.116
Grid	19	100 / 100	2.058	4.398	100 / 100	2.722	5.295	96	-4.10415	0.248
	20	100 / 100	0.043	0.015	100 / 100	0.032	0.013	100	-3.34002	< 0.001
	21	94 / 100	10.023	23.107	100 / 100	5.867	10.676	54	-2.12963	0.054
Star	22	100 / 100	38.584	150.159	100 / 100	196.849	526.998	0	-	0.006
	23	100 / 100	0.645	4.805	100 / 100	16.689	106.388	0	-	0.131
	24	79 / 100	155.238	503.811	100 / 100	107.067	252.888	2	0.00002	0.305

5.3 Shortest Path Diverting

As with the maximum flow NDP model, the shortest path formulation is partitioned into the leader's problem and the follower's problem. The leader's problem is as described in the MNDP-a model. The follower's problem can be formulated using the typical shortest path model [6:p. 607]. The follower's decision variable x_{ij} indicates whether arc (i, j) is along the shortest source-terminus path.

The follower's objective is combined with the leader's by adding it, thus preserving its minimization:

$$\underbrace{\sum_{(i,j)\in A} c_{ij}y_{ij}}_{\text{(leader)}} + \lambda \underbrace{\sum_{(i,j)\in A} d_{ij}x_{ij}}_{\text{(follower)}}$$

The magnitude of the weight of the follower's objective λ is a cooperation factor small enough to ensure leader preference, as in the MNDP-a model.

Following is the formulation for the shortest path network diverting problem for arcs only (**SNDP-a**):

$$\min \quad \sum_{(i,j)\in E} c_{ij}y_{ij} + \lambda \sum_{(i,j)\in E} d_{ij}x_{ij} \tag{5.2a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.2b}$$

$$u_s = 0, \tag{5.2c}$$

$$u_k = 1, \qquad \forall k \in D \subset V, \tag{5.2d}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V, \tag{5.2e}$$

$$x_{ij} \leq 1 - y_{ij}, \qquad \forall (i,j) \in E, \tag{5.2f}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.2g}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.2h}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{5.2i}$$

The objective (5.2a) ensures that the leader's minimum cost s-D cut set is selected while minimizing the shortest source-to-terminus path on the residual network. The cost to interdict arc (i,j) and the distance from node i to node j are represented by c_{ij} and d_{ij}, respectively.

The leader's (attacker's) problem consists of the first term of the objective function and constraints (5.2b)-(5.2d) and (5.2h)-(5.2i) and are as described for the MNDP-a model.

The follower's problem is the shortest path problem and consists of the last term in the objective function and constraints (5.2e)-(5.2g). Constraints (5.2e) ensure flow conservation at each node, a single unit of flow originating from the source, and a single unit of flow

terminating at the destination. Constraint (5.2f) ensures the arc is not on the shortest path if the leader interdicts the arc. Notice that there is not a feasible solution for the leader that interdicts all paths from the source to the terminus due to constraints (5.2e).

Because the leader's problem is unchanged from the MNDP-a model, the worst-case bound for the interdiction cost is the sum of the interdiction costs of all arcs entering (or leaving) the divert set.

5.3.1 Numerical Examples.

Numerical examples illustrate the SNDP-a model. Consider again the directed network depicted in Figure 19 where each arc has a unit interdiction cost and distance, and with shaded divert set $D = \{2, 4, 6\}$. The optimal solution of the SNDP-a model is the leader's minimum cut of 2 (arcs {(1,2), (3,4)}) with a follower's shortest path of 4 (*i.e.*, $1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 8$).

Next, consider the United States Interstate System. A map [60] of the major US interstates was used to place nodes at the interstate intersections and the arc distances were estimated using Google maps. The resulting network consists of 143 nodes with 251 undirected arcs (502 directed) and is depicted in Figure 21. Consider a shipping company that seeks to transport goods from San Diego (node 8) to Miami (node 117). The shortest path route traverses through the southern most nodes to Jacksonville (node 114) and then proceeds south to Miami as indicated by the highlighted route in Figure 21. Suppose that a natural disaster, such as Hurricane Katrina in 2005, has caused damage and traffic congestion in the shaded region covering the southern United States. The shipping company would then desire a shortest path that diverts from the damaged and congested roads. The SNDP-a model solves this problem utilizing a divert set consisting of nodes {41, 48, 49, 50, 64, 65, 66, 67, 68, 69, 75, 76, 77}. The solution is for the company to direct drivers to avoid travel along the arcs indicated with an X and follow the route highlighted in Figure 22. The new

shortest path is 3,050 miles whereas the pre-diverting path was 2,686 miles. The shortest path increased by nearly 400 miles, but satisfies the decision criteria.

5.3.2 Testing and Results.

The SNDP-a model is tested on the same test network instances that were utilized to test the MNDP-a model. Table 47 lists the parameters utilized to generate the 10 replicates for each of the 24 network structures listed. Recall that 10 unique divert sets were constructed for each replicate.

Table 51 lists the results of the model that utilizes s-D cuts and a value of either $\lambda = 0.00001$ or $\lambda = 0$. For each of the solution approaches, the number of instances solved to optimality, mean solution time, and solution time standard deviation are recorded. In addition, the number of network instances that resulted in unequal solution values is tabulated. The average difference, if one exists, is also included. Finally, for the network instances tested, the two-tailed paired-t statistic is utilized to determine whether there is a difference in the solution times when $\lambda = 0.00001$ or $\lambda = 0$.

The reason that the number of network instances solved to optimality is not 100/100 for every test is that the divert set was constructed by selecting a random set of connected nodes. In some structures tested, the divert sets were large enough to be a cut set. In other words, when no node in the divert set is traversed, no s-t path exists.

A two-tailed paired-t test statistic is utilized to determine whether there is a difference in the solution times when $\lambda = 0.00001$ or $\lambda = 0$. In cases when the p-value is less than 0.05, there is sufficient evidence at the 0.05 level of significance to reject the claim that there is no difference in the solution times. For larger p-values, there is insufficient evidence at the 0.05 level of significance to reject the claim of no difference in the solution times.

Examination of the p-values comparing $\lambda = 0.00001$ and $\lambda = 0$ for mean solution times are larger than 0.05 for all network cases tested except the cases 4, 6, and 17. Within these tests

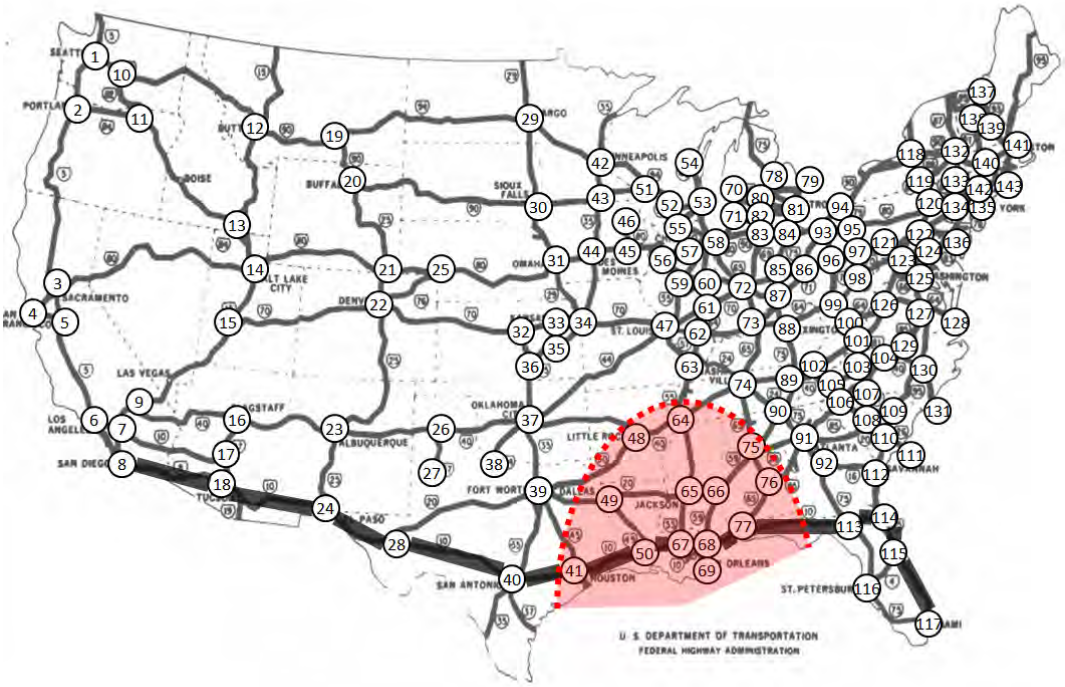


Figure 21. Shortest path from San Diego to Miami and effects of Hurricane Katrina

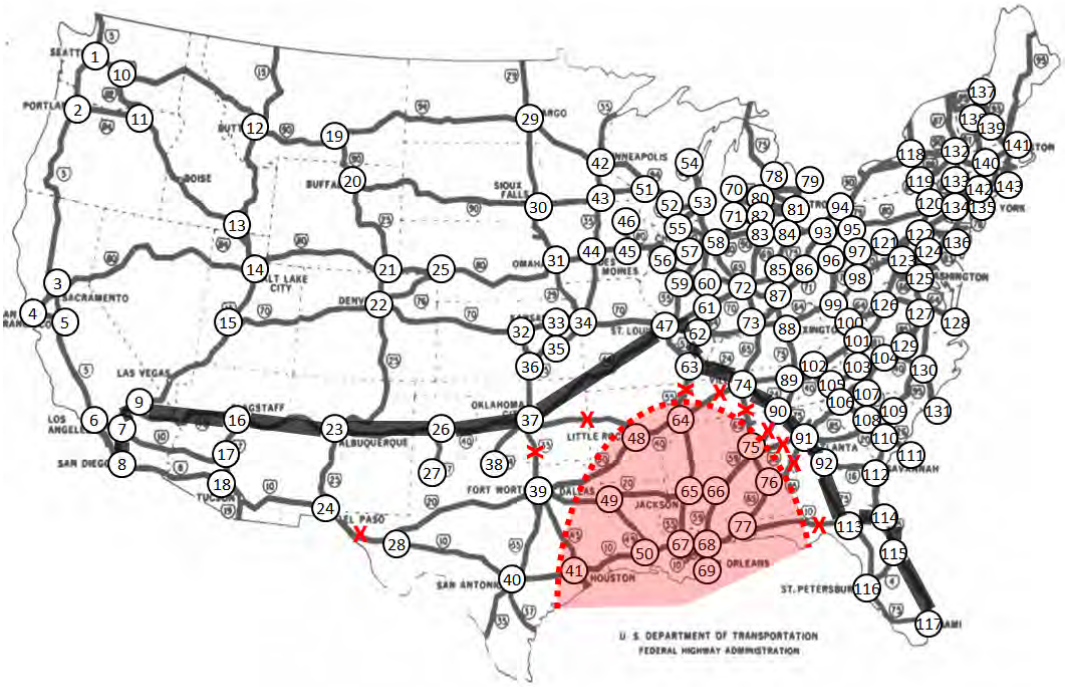


Figure 22. US Interstate System example diverting due to Hurricane Katrina

cases, two favor the $\lambda = 0.00001$ solution and the other favors $\lambda = 0$. There is insufficient evidence at the 0.05 level of significance to reject the claim of no difference in the solution times for the network instances tested. Therefore, the decision to use either value remains arbitrary. The follower's term could be removed from the objective without impacting the solution times for the network instances tested. This did not result in a change in the arc cut set solutions because the constraints ensure the existence of a follower's shortest path after paths to the divert set have been diverted.

In network cases 13 and 14, the SNDP-a model was infeasible for nine of the ten networks tested. Recall that each of ten network topologies were tested with ten different divert set instances. In the one network that resulted in optimal solutions for shortest path diverting, the network had a mean degree of 3.4. The nine test networks that yielded infeasible solutions each had mean degrees of less than 3. This indicates that the networks tested were sparse and may indicate a possible threshold at which network diverting is feasible. Further testing with very sparse networks with various mean degree values may address this shortcoming. Smaller mean degree values indicate that the network topology may be a tree. A tree topology is disconnected by the removal of a single node, which further informs the findings in the networks tested.

Table 51. Comparison of results for s-D cuts with $\lambda = 0.00001$ and $\lambda = 0$

Structure	Case	$\lambda = 0.00001$			$\lambda = 0$			Solutions		2-Tail
		# Solved	Mean Time	StDev	# Solved	Mean Time	StDev	# $\neq$	Mean Change	p-value
ER	1	100 / 100	7.722	15.555	100 / 100	8.034	17.320	0	-	0.568
	2	100 / 100	15.289	24.543	100 / 100	14.309	23.921	0	-	0.149
	3	100 / 100	88.540	104.958	100 / 100	88.780	103.721	0	-	0.947
	4	100 / 100	97.604	85.574	100 / 100	111.338	86.354	0	-	0.026
BA	5	100 / 100	1.875	15.573	100 / 100	1.696	13.226	0	-	0.475
	6	100 / 100	2.670	9.599	100 / 100	3.139	11.666	0	-	0.050
	7	100 / 100	1.070	3.417	100 / 100	1.454	4.700	0	-	0.151
	8	100 / 100	37.254	259.286	100 / 100	45.118	337.923	0	-	0.326
WS	9	100 / 100	1.644	8.841	100 / 100	1.349	6.329	0	-	0.257
	10	100 / 100	1.936	4.378	100 / 100	1.871	4.291	0	-	0.467
	11	100 / 100	78.268	107.247	100 / 100	80.515	115.413	0	-	0.589
	12	100 / 100	40.945	34.527	100 / 100	42.950	37.614	0	-	0.523
PNDCG	13	10 / 100	0.034	0.048	10 / 100	0.028	0.035	0	-	0.275
	14	8 / 100	0.203	0.267	8 / 100	0.072	0.045	0	-	0.150
	15	70 / 100	0.607	1.028	70 / 100	0.542	0.961	0	-	0.109
	16	78 / 100	1.056	5.153	78 / 100	0.982	4.539	0	-	0.474
	17	100 / 100	10.982	10.517	100 / 100	9.476	7.985	0	-	0.021
	18	100 / 100	15.918	35.479	100 / 100	16.433	36.781	0	-	0.507
Grid	19	100 / 100	0.700	1.702	100 / 100	0.702	1.459	0	-	0.970
	20	100 / 100	0.029	0.014	100 / 100	0.030	0.012	0	-	0.565
	21	94 / 100	3.922	7.518	94 / 100	3.548	6.218	0	-	0.047
Star	22	100 / 100	18.527	78.282	100 / 100	16.949	57.679	0	-	0.510
	23	100 / 100	0.250	1.596	100 / 100	0.243	1.528	0	-	0.421
	24	79 / 100	81.898	245.506	79 / 100	83.075	234.712	0	-	0.821

5.4 Model Extensions

5.4.1 Extending the Divert Set.

The formulation of the leader's problem in the MNDP-a (5.1) and SNDP-a (5.2) models assumes that the divert set D consists only of nodes and that they will not be interdicted in the problem's solution. However, the scenario or instance of the problem being modeled may be more suited to a divert set consisting of arcs only. It is also possible that a divert set be allowed to contain both nodes and arcs.

In the case that the divert set should contain arcs only, the arc in the divert set could be transformed into a node. To transform the arc, it is replaced by two arcs and a node as in the transformation shown in Figure 23. The capacity of arc (i, j) is assigned to both new arcs and their interdiction cost is large enough that they will not be interdicted. They cannot be targeted since the leader interdicts arcs by assumption and the two new arcs should not be targeted since they are part of the divert set. This arc splitting procedure allows node i to be interdicted if nodal interdiction is permitted and does not unnecessarily restrict the divert set to nodes that should not be included if the tail or head nodes of the arc were added as a proxy for the arc in the divert set.

If the divert set is allowed to contain both nodes and arcs, the arc splitting procedure should be used for arcs in the divert set unless the node at the head or tail of the arc is also in the divert set. Then the split is redundant since flow on the arc will be interdicted because solution to the NDP will permit no flow into the divert set. This transformation will convert any arcs to nodes without unnecessarily restricting the flow on nodes and arcs adjacent to the divert set. The increase in the number of decision variables will depend on

Figure 23. Arc (i, j) is split and proxy node i' takes its place

the number of arcs converted to nodes. If the number of arcs in the divert set is small, the resulting increase in the number of variables will be minimal. However, the problem size can become too large with the addition of many arcs to the divert set.

5.4.2 Model Extensions for NDP.

The network diverting problem can be extended to account for several modifications. Without loss of generality, the maximum flow or shortest path network diverting formulation will be used to illustrate the changes. Each of the modifications are implemented individually, unless noted otherwise. Multiple changes can be incorporated in a single model if necessary by making each of the appropriate modifications.

The first extension is to allow more than one disjoint divert set. Let L be the index for each disjoint divert set. Thus, the set of divert nodes is $D = \bigcup_{\ell \in L} D_\ell$. The formulation will then ensure that no flow is permitted to all divert sets for the leader's problem and will determine either the maximum flow remaining or the shortest path for the follower. The leader's decision variable is indexed for the multiple divert sets and becomes y_{ij}^{ℓ}. It indicates whether the flow from the source to the ℓth divert set is interdicted along arc (i, j). The indicator variable for which side of the cut is also indexed by ℓ. The formulation for the shortest path network diverting problem for arcs only with multiple divert sets (**SNDP-aM**)

follows:

$$\min \quad \sum_{\ell \in L} \sum_{(i,j) \in E} (c_{ij} y_{ij}^{\ell}) + \lambda \sum_{(i,j) \in E} (d_{ij} x_{ij}) \tag{5.3a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij}^{\ell} \geq 0, \qquad \forall (i,j) \in E, \forall \ell \in L, \tag{5.3b}$$

$$u_s^{\ell} = 0, \qquad \forall \ell \in L, \tag{5.3c}$$

$$u_k^{\ell} = 1, \qquad \forall k \in D_{\ell} \subset V, \forall \ell \in L, \tag{5.3d}$$

$$\sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V, \tag{5.3e}$$

$$x_{ij} \leq 1 - y_{ij}^{\ell}, \qquad \forall (i,j) \in E, \forall \ell \in L, \tag{5.3f}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.3g}$$

$$y_{ij}^{\ell} \in \{0,1\}, \qquad \forall (i,j) \in E, \forall \ell \in L, \tag{5.3h}$$

$$u_i^{\ell} \in \{0,1\}, \qquad \forall i \in V, \forall \ell \in L. \tag{5.3i}$$

In this model, all constraints are as described for the SNDP-a model. The index set for each of the divert sets is included in the objective (5.3a) and Constraints (5.3b)-(5.3d), (5.3f), and (5.3h)-(5.3i). The addition of multiple divert sets does not alter the follower problem or the linking Constraint (5.3f) since the flow on the shortest path will not be allowed if the arc is interdicted for any of the possible source to divert set cuts.

Next, nodes are allowed to be interdicted to divert flow from the divert set. For node interdiction, a decision variable v_i is introduced where $v_i = 1$ if node i is interdicted and $v_i = 0$ otherwise, similar to the nodal interdiction of Kennedy *et al.* [45]. This leads to a

formulation for the maximum flow network diverting problem for nodes only (**MNDP-n**):

$$\min \quad \sum_{i \in V} (c_i v_i) - \lambda x_{ts} \tag{5.4a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.4b}$$

$$u_s = 0, \tag{5.4c}$$

$$u_k = 1, \qquad \forall k \in D \subset V, \tag{5.4d}$$

$$y_{ij} = v_i, \qquad \forall (i,j) \in E, \tag{5.4e}$$

$$v_\ell = 0, \qquad \ell = s, \ell = t, \tag{5.4f}$$

$$\sum_{j:(i,j) \in E'} x_{ij} - \sum_{j:(j,i) \in E'} x_{ji} = 0, \qquad \forall i \in V, \tag{5.4g}$$

$$x_{ij} \leq b_{ij}(1 - y_{ij}), \qquad \forall (i,j) \in E, \tag{5.4h}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.4i}$$

$$x_{ts} \geq 1, \tag{5.4j}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.4k}$$

$$v_\ell \in \{0,1\}, \qquad \forall \ell \in V, \tag{5.4l}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{5.4m}$$

In this model, all constraints are as described for the MNDP-a model. The objective accumulates the cost to the leader of interdicting nodes (c_i represents the cost to interdict node i) while maximizing flow for the follower. The addition of Constraint (5.4e) enforces the cut of outgoing arcs from the interdicted node. Constraint (5.4f) ensures that the source and terminus are not interdicted. For D-t cuts, the formulation is changed slightly to account for not interdicting nodes in the divert set, *i.e.* Constraint (5.4e) becomes $y_{ij} = v_j, \forall (i,j) \in E$.

To modify the MNDP-n to allow both node and arc interdiction, the only change is to a single constraint. Constraint (5.4e) becomes $y_{ij} \leq v_i, \forall (i,j) \in A$. This allows for either

arc or node interdiction. As before, the constraint is modified for D-t interdiction to be $y_{ij} \leq v_j, \forall (i,j) \in A$. Therefore, it is necessary to 'refund' the arc interdiction cost as a result of targeting a node [45]. Thus, the maximum flow formulation of the network diverting problem for both arcs and nodes (**MNDP-b**) is:

$$\min \quad \left(\sum_{(i,j)\in E} (c_{ij}(y_{ij} - v_i)) + \sum_{i \in V} c_i v_i \right) - \lambda x_{ts} \tag{5.5a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.5b}$$

$$u_s = 0, \tag{5.5c}$$

$$u_k = 1, \qquad \forall k \in D \subset V, \tag{5.5d}$$

$$y_{ij} \geq v_i, \qquad \forall (i,j) \in E, \tag{5.5e}$$

$$v_\ell = 0, \qquad \ell = s, \ell = t, \tag{5.5f}$$

$$\sum_{j:(i,j)\in E'} x_{ij} - \sum_{j:(j,i)\in E'} x_{ji} = 0, \qquad \forall i \in V, \tag{5.5g}$$

$$x_{ij} \leq b_{ij}(1 - y_{ij}), \qquad \forall (i,j) \in E, \tag{5.5h}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.5i}$$

$$x_{ts} \geq 1, \tag{5.5j}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.5k}$$

$$v_i \in \{0,1\}, \qquad \forall i \in V, \tag{5.5l}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{5.5m}$$

The leader may be successful in diverting if the path or flow through the divert set is lower than a certain threshold. This may be the case when the leader determines that the follower will not be able to overcome the leader in the future due to the diminished path or flow. This situation can be incorporated in the diverting models by setting a value in a constraint that limits flow or path in the divert set. The formulation for the shortest path

network diverting problem for arcs only with a divert set threshold (**SNDP-aT**) follows:

$$\min \quad \sum_{(i,j)\in E} c_{ij}y_{ij} + \lambda \sum_{(i,j)\in E} d_{ij}x_{ij} \tag{5.6a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.6b}$$

$$u_s = 0, \qquad \forall i \in V, \tag{5.6c}$$

$$u_k = 1, \qquad \forall k \in D \subset V, \tag{5.6d}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V, \tag{5.6e}$$

$$x_{ij} \leq 1 - y_{ij}, \qquad \forall (i,j) \in E, \tag{5.6f}$$

$$\sum_{\substack{i \text{ and/or } j\in D: \\ (i,j)\in E}} d_{ij}x_{ij} \leq T, \tag{5.6g}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.6h}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{5.6i}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{5.6j}$$

The objective and constraints are as described for the SNDP-a model. Constraint (5.6g) ensures the total distance the follower can traverse within the divert set is less than the threshold T.

Each of the model extensions presented in this section can be applied individually or combined in a single formulation to increase the realism of the model.

The MNDP-a formulation is modified to account for a case where there are not enough resources to completely halt flow to the divert set. To facilitate this change, a partition of the cut into those selected and those not actually interdicted but still in the cut set similar

to the method Wood [68] is used. The variable z_{ij} is added which denotes whether arc (i, j) is in the s-D cut but not targeted for attack. The amount of interdiction resources required to cut the flow on an arc is denoted r_{ij} and the amount the attacker has available is R. This gives the network diverting problem for arcs only with resource constraints (**MNDP-aR**).

$$\min \quad \left(\sum_{(i,j)\in E} (b_{ij} z_{ij}) - \varepsilon w \right) - \lambda x_{ts} \tag{5.7a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} r_{ij} y_{ij} + w = R, \qquad \forall (i,j) \in E, \tag{5.7b}$$

$$u_i - u_j + z_{ij} + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.7c}$$

$$u_s = 0, \tag{5.7d}$$

$$u_k = 1, \qquad \forall k \in D \subset V, \tag{5.7e}$$

$$\sum_{j:(i,j)\in E'} x_{ij} - \sum_{j:(j,i)\in E'} x_{ji} = 0, \qquad \forall i \in V, \tag{5.7f}$$

$$x_{ij} \leq b_{ij}(1 - y_{ij}), \qquad \forall (i,j) \in E, \tag{5.7g}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{5.7h}$$

$$x_{ts} \geq 1, \tag{5.7i}$$

$$y_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in E, \tag{5.7j}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in E, \tag{5.7k}$$

$$u_i \in \{0, 1\}, \qquad \forall i \in V. \tag{5.7l}$$

In this model, the objective function value is penalized for not selecting a cut that would potentially allow flow to the divert set. As the amount of interdiction resources diminish, the model is forced to allow flow to the divert set and the first part of the objective gives a worst-case flow through the divert set. The follower might not select that specific flow based on the selection of multiple maximum flow solutions, but flow through D up to a value of

$\sum_{(i,j)\in E} (c_{ij} z_{ij})$ is possible. Constraints (5.7b), (5.7c), and (5.7k) ensure the desired partitioning of the s-D cut into those arcs interdicted and those that are not.

Constraint (5.7b) introduces a slack-like variable w representing the residual resource (equivalent to unused resource), and a multiple of this residual is included in the objective function to reward the conservation of resources (similar to Kallemyn *et al.* [44]). Because the residual is largest when conserving the resource, an ε-multiple of the residual resource is added in the minimizing objective. The ε should be small enough that the magnitude of the actual network flow is not altered, but not so small that the effect of resource conservation is lost. In effect, the use of this term in the objective acts as a penalty function on the expenditure of interdiction resources and discriminates among otherwise alternative optimal solutions, if any exist.

5.5 Summary

This chapter described and developed the network diverting problem. The contributions are summarized in Table 52. Maximum flow and shortest path network diverting models were formulated and solved over many test instances. The timing results for the test instances were analyzed. The largest factor in time to solve the NDP for the network instances tested is whether the divert set is more restrictive than the unconstrained maximum flow or shortest path solution. The relationship between diverting and channeling was also examined. The next chapter considers the impact of disrupting a network.

Table 52. Diverting Interdiction Task Contributions

Interdiction Task	Section	Contribution	Description
Divert	5.2	MNDP-a	The maximum flow network diverting problem for arcs only determines the leader's minimum cost source-to-divert set cut while maximizing the remaining source-to-terminus flow on the network for the follower.
	5.3	SNDP-a	The shortest path network diverting problem for arcs only determines the leader's minimum cost source-to-divert set cut to ensure no path for the follower traverses the divert set.
	5.4	SNDP-aM	Extends SNDP-a model. The model represents diverting shortest paths from multiple divert sets.
	5.4	MNDP-n	Extends MNDP-a model. The model represents diverting the follower's residual flow from the divert set by targeting nodes.
	5.4	MNDP-b	Extends MNDP-a model. The model represents diverting the follower's residual flow from the divert set by targeting arcs and/or nodes.
	5.4	SNDP-aT	Extends SNDP-a model. The model represents targeting arcs such that the total distance traversed within the divert set is less than a threshold.
	5.4	MNDP-aR	Extends MNDP-a model. The model represents diverting the residual flow from the divert set with limited resources.

VI. Disrupt Interdiction Tasks

6.1 Introduction

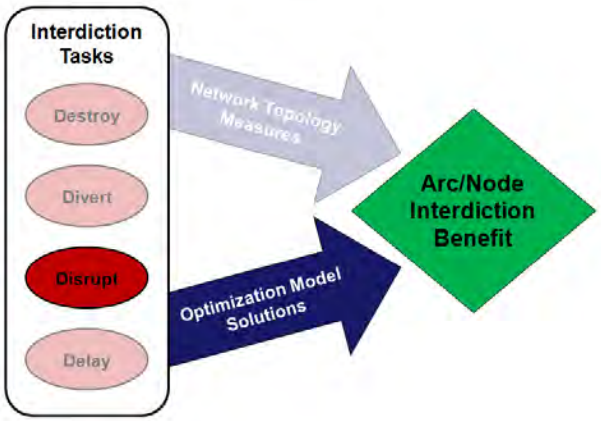


Figure 24. Research Framework: Network Disrupting Models

This chapter addresses the models approach dealing with the 'disrupt' interdiction task depicted in the research framework depicted in Figure 24. In this research, the term 'disrupt' refers to arcs or nodes targeted for interdiction that are not destroyed, but rather disrupted in some way to alter their capacity. This chapter develops a mathematical programming modeling framework for disrupting a network's capabilities based on a fixed probability of inflicting intended damage against each targeted arc and/or node.

Whenever a physical object is targeted and attacked, there is some probability that the attack does not completely destroy the target, by design or otherwise. Given the stochastic nature of this phenomenon, modeling various levels of destruction in the network and the resulting disruptions will significantly increase the realism of these models. In assessing battle damage, analysts must make determinations about whether a target sustained damage and, if so, whether it is still functional. These measurements are based on the analyst's perception of what percentage of the network's capability remains.

The level of damage that occurs as a result of an interdiction attack is not known. Consider two parameters that indicate the impact to an arc when it is targeted. First, let q_{ij} denote the percent increase in the length (time) required to traverse arc (i, j). This parameter is utilized when the follower's objective is minimizing the shortest path length between a source and terminus. Second, let r_{ij} represent the percent reduction for the capacity of arc (i, j). This parameter is bounded between 0 and 1 and is utilized when the follower seeks to maximize network flow. The interdiction strategy is determined based on the flow or path across the residual network after interdiction. In this research, when dealing with battle damage on a flow network, if an attacker desires to use the actual probability of kill p_k, its value is used as the reduction.

Additionally, arcs may be targeted for disruptive attacks that do not involve weaponry. Within the cyber domain, a disruptive attack may increase the time for a message to traverse an arc without alerting the network operator that the arc is compromised. Meanwhile, the attacker may complete their mission before a command message reaches troops or, alternatively, a sensor alert message reaches the decision maker operating the targeted network.

6.2 Disrupting Paths and Flows

In a manner similar to that of Israeli and Wood [42], the arc's length or capacity is increased if the arc is targeted. Whereas Israeli and Wood increased the arc's distance by an integer value, as described in Section 2.3.1, the proposed disruptive models increase (decrease) the arc's length (capacity) by a percentage of the arc's distance-weight (capacity-weight).

For the disrupting models developed in this chapter, several decision variables will be commonly utilized. The binary y_{ij} variable indicates whether arc (i, j) is targeted for interdiction disrupting tasks. Typically, the models utilize a cut set to identify the arcs that take a value of $y_{ij} = 1$ when targeted. The cut set is identified via the variable u_i which

assigns nodes to respective sets indicated by a 0 or 1 value for the u_i-variable. The cut set denotes the arcs that connect these sets. In some instances, every arc in the cut set may not be targeted. In such cases, the variable z_{ij} indicates whether arc (i, j) is contained in the cut set but not targeted. Because the models represent the decisions of a leader (who takes action based on y and z) and a follower, the final variable represents the decision of the follower: the variable x_{ij} indicates the flow across arc (i, j) as determined by the follower through the residual network. The x-variable does not impact the values of the leader's variables, but it is maintained in the models to account explicitly for the follower's actions. In flow models where the objective function identifies the minimum capacity cut set, the follower's problem can be removed because the objective explicitly computes the follower's flow. In flow models having an alternate objective, such as minimizing cost, the follower's maximum flow problem is retained.

Let d_{ij} represent the length of the arc from node i to node j. The value of d_{ij} can also represent the travel time along the arc or other additive measures that can be minimized.

The formulation for the disruptive shortest path problem for arcs only (**DSP-a**):

$$\min \quad \sum_{(i,j)\in E} (c_{ij} y_{ij}) + \lambda \sum_{(i,j)\in E} x_{ij} d'_{ij} \tag{6.1a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{6.1b}$$

$$u_s = 0, \tag{6.1c}$$

$$u_t = 1, \tag{6.1d}$$

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V, \tag{6.1e}$$

$$d'_{ij} = d_{ij}(1 + y_{ij} q_{ij}), \qquad \forall (i,j) \in E, \tag{6.1f}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E \tag{6.1g}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{6.1h}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{6.1i}$$

The objective (6.1a) can be separated into the leader's and the follower's objectives. The leader minimizes interdiction cost. The follower minimizes the shortest path which is determined by the expected increase in arc distances based on damage incurred as a result of the interdiction strategy in the last term of (6.1a). If an arc is not interdicted, the value of d'_{ij} is unchanged from d_{ij} when computing the length of the shortest path. If an arc is interdicted, the distance is increased by q percent. A linking Constraint (*i.e.*, $x_{ij} \leq 1 - y_{ij}, \forall (i,j) \in E$) is not included since the path can still be selected even if the arc is targeted as part of the interdiction strategy. In other words, a targeted arc can be used by the network operator, but it might be degraded. All other constraints are as described for the SNDP-a formulation.

Note that the DSP-a model is nonlinear. The follower's portion of the objective contains the product of the x- and y-variables when the value of d'_{ij} is substituted from Constraint 6.1f. Because the DSP-a model is nonlinear (as opposed to all the other models developed throughout this research, which are linear), it is not tested. Additionally, the DSP-a model can be solved utilizing the methodology of Israeli and Wood [42] since its form is similar in structure.

Alternatively, a disrupting task may be planned against a maximum flow network. Whereas the shortest path disrupting model is similar to one found in the open literature, no maximum flow disrupting model was found. Let b_{ij} denote the capacity of arc (i, j). The formulation for the disruptive maximum flow problem for arcs only (**DMP-a**):

$$\min \quad \sum_{(i,j)\in E} (b_{ij}(1 - r_{ij})y_{ij}) - \lambda x_{ts} \tag{6.2a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{6.2b}$$

$$u_s = 0, \tag{6.2c}$$

$$u_t = 1, \tag{6.2d}$$

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 0, \qquad \forall i \in V, \tag{6.2e}$$

$$x_{ij} \leq b_{ij}(1 - y_{ij}r_{ij}), \qquad \forall (i,j) \in E, \tag{6.2f}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{6.2g}$$

$$y_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in E, \tag{6.2h}$$

$$u_i \in \{0, 1\}, \qquad \forall i \in V. \tag{6.2i}$$

Again, the objective (6.2a) is partitioned into two parts: one for the leader and one for the follower. The leader minimizes residual network capacity. The follower maximizes the expected flow which is determined by the expected residual capacity of the arcs as a result of damage incurred during interdiction in Constraint (6.2f). If an arc is not interdicted, the

value of x_{ij} is bounded above by the capacity b_{ij}. If an arc is targeted, the value of x_{ij} is bounded above by the reduced capacity. All other constraints are as described for the MNDP-a formulation.

The objective for the DMP-a model selects a minimum capacity cut set within the residual network. Alternatively, the objective function could be

$$\sum_{(i,j)\in E} (c_{ij}y_{ij}) - \lambda x_{ts}.$$

This form of the objective minimizes the cost of targeting arcs and disrupts source-terminus flow. The two objectives considered yield the minimum capacity and cost impacts from the leader's perspective, respectively.

The weights for the objective functions of each of the disruptive models (*i.e.*, (6.1a) and (6.2a)) correspond to the importance of the follower's decision. The value of the weight λ of the follower's problem should be small enough to ensure that the leader has preemptive decision preference. To ensure that the leader has preemptive preference over the follower, the weight for λ can be determined by utilizing Sherali's Algorithm 1 [58].

6.2.1 Notional Example.

Consider again the notional military transportation network example from Ghare *et al.* [33] and Wood [68] as depicted in Figure 25. The capacity, interdiction costs, and expected percent capacity reduction as a result of being targeted are enumerated in Table 53. Two separate damage estimates are given to demonstrate the impact of differences in the capacity reduction as a result of striking arcs.

For the first set of expected reduction levels (within the column labeled $(r_{ij})_1$) using the minimum capacity cut objective, the solution of DMP-a model yields a cut set of arcs {(2,9), (2,6), (2,7), (3,6), (3,7), (3,8), (4,7), (4,8), (5,7), (5,8), (5,12)} for a cost of 47 units of

Arc	Capacity	Cost	$(r_{ij})_1$	$(r_{ij})_2$	Arc	Capacity	Cost	$(r_{ij})_1$	$(r_{ij})_2$
(1,2)	1000	100	0.00	0.00	(6,10)	60	7	0.25	0.75
(1,3)	1000	100	0.00	0.00	(7,10)	120	4	0.25	0.75
(1,4)	1000	100	0.00	0.00	(7,11)	150	6	0.25	0.75
(1,5)	1000	100	0.00	0.00	(8,11)	120	6	0.25	0.75
(2,6)	60	5	0.50	0.67	(8,12)	80	4	0.25	0.75
(2,9)	70	4	0.25	0.75	(9,13)	80	4	0.33	0.80
(2,7)	60	5	0.50	0.67	(9,14)	50	5	0.33	0.80
(3,6)	50	3	0.50	0.67	(10,13)	100	5	0.33	0.80
(3,7)	50	3	0.50	0.67	(10,14)	80	4	0.33	0.80
(3,8)	60	5	0.50	0.67	(11,14)	180	6	0.33	0.80
(4,7)	100	3	0.50	0.67	(11,15)	100	4	0.33	0.80
(4,8)	80	5	0.50	0.67	(12,14)	80	5	0.33	0.80
(5,7)	50	5	0.50	0.67	(12,15)	100	6	0.33	0.80
(5,8)	100	5	0.50	0.67	(13,16)	1000	100	0.00	0.00
(5,12)	80	4	0.25	0.75	(14,16)	1000	100	0.00	0.00
(6,9)	60	4	0.25	0.75	(15,16)	1000	100	0.00	0.00

Table 53. Data for the notional military transportation network in Figure 25 [33, 68]

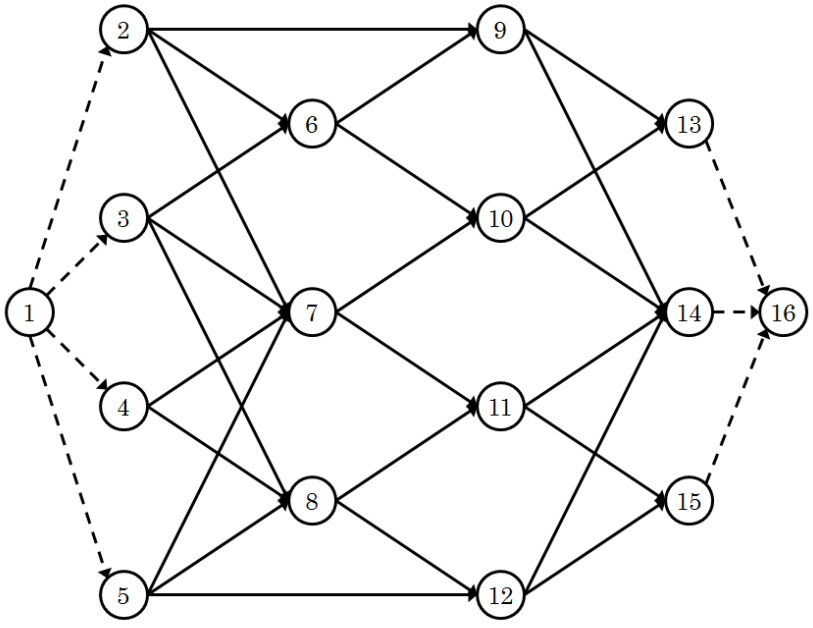


Figure 25. A notional military transportation network [33, 68]

interdiction resource and allows a maximum flow across the residual network of 417.5 units. When considering the expected reduction damage levels of $(r_{ij})_2$, the cut set consists of arcs {(9,13), (9,14), (10,13), (10,14), (11,14), (11,15), (12,14), (12,15)} for a cost of 39 units of interdiction resource and allows a maximum flow across the residual network of 154 units.

When the objective is modified to compute the minimum cost cut set, the cut set consists of arcs {(2,9), (2,6), (3,6), (7,10), (8,12), (11,14), (11,15), (5,12)} for a cost of 34 units of interdiction resource regardless of capacity reduction levels. The residual flow as a result of disrupting network attacks with damage levels that reduce capacities according to the table are 504.167 and 180.167 units for reduction levels $(r_{ij})_1$ and $(r_{ij})_2$, respectively.

In these notional example network instances, the two objectives (*i.e.*, minimize cost and minimize capacity cut) yield solutions that demonstrate to the decision maker the trade-off between interdiction cost and residual flow.

6.2.2 Testing and Results.

The DMP-a model is tested on network instances that are generated as described in Appendix A. The parameters for each network structure utilized are given in Table 54. For each of the 24 network structure cases listed, 30 replicates were generated.

The values of the interdiction cost-weight and the capacity-weight assigned to each arc in the network are selected using a discrete (integer) uniform distribution having a range between 1 and 10, inclusively. The reduction in capacity achieved by targeting an arc is determined via a uniform distribution assigning r_{ij} to one of three values: 0.25, 0.50, or 0.75.

The DMP-a model is solved for the various random network instances using IBM® ILOG® CPLEX® Optimization Studio V12.6. The solutions when the DMP-a model is solved are displayed in Table 55.

Table 54. Parameter settings for 100-node random network structures

Structure	Parameters	Case	Mean Density
ER	$\beta = 0, p = 0.1$	1	0.098
	$\beta = 1, p = 0.1$	2	0.099
	$\beta = 0, p = 0.5$	3	0.500
	$\beta = 1, p = 0.5$	4	0.502
BA	$m_a = 2, n_0 = 25$	5	0.091
	$m_a = 5, n_0 = 25$	6	0.136
	$m_a = 2, n_0 = 55$	7	0.318
	$m_a = 10, n_0 = 50$	8	0.348
WS	$k = 4, p = 0.10$	9	0.040
	$k = 4, p = 0.25$	10	0.040
	$k = 30, p = 0.10$	11	0.303
	$k = 30, p = 0.25$	12	0.303
PNDCG	$\beta = 0, \alpha = 2.35$	13	0.025
	$\beta = 1, \alpha = 2.35$	14	0.025
	$\beta = 0, \text{dist} = U_{3,5}$	15	0.030
	$\beta = 1, \text{dist} = U_{3,5}$	16	0.030
	$\beta = 0, \text{dist} = U_{20,5}$	17	0.202
	$\beta = 1, \text{dist} = U_{20,5}$	18	0.202
Grid	10×10	19	0.037
	5×20	20	0.038
	20×5	21	0.035
Star	10×10	22	0.040
	5×20	23	0.040
	20×5	24	0.040

The number solved shows the number of network instances that resulted in an optimal solution within the 3,600 second time limit imposed on the solver. Every network instance tested yielded an optimal solution.

Network density is graphed against solution times in Figure 26 for the network instances tested. The line represents a linear approximation of the relationship between density and solution time. For the networks tested, solution time increases with an associated increase in network density. The coefficient of determination R^2 between density and solution time, given in parentheses in the legend of the chart for the tested network cases, is 0.887. The

Table 55. Results for DMP-a for 100-node network instances

Structure	Type	# Solved	Mean Time	StDev Time	Mean Its	StDev Its
ER	1	30 / 30	0.063	0.006	284.3	57.508
	2	30 / 30	0.063	0.006	298.3	71.723
	3	30 / 30	0.385	0.161	577.6	65.225
	4	30 / 30	0.355	0.019	596.3	57.262
BA	5	30 / 30	0.053	0.008	39.8	24.814
	6	30 / 30	0.078	0.005	120.5	44.573
	7	30 / 30	0.186	0.005	96.5	61.612
	8	30 / 30	0.215	0.017	228.5	33.823
WS	9	30 / 30	0.030	0.006	220.3	44.852
	10	30 / 30	0.031	0.005	209.3	38.909
	11	30 / 30	0.211	0.008	559.2	59.465
	12	30 / 30	0.209	0.009	495.6	35.764
PNDCG	13	30 / 30	0.015	0.008	71.1	37.016
	14	30 / 30	0.019	0.006	74.9	35.740
	15	30 / 30	0.023	0.008	121.2	51.401
	16	30 / 30	0.023	0.008	132.8	41.752
	17	30 / 30	0.130	0.009	396.7	32.747
	18	30 / 30	0.130	0.012	401.3	36.535
Grid	19	30 / 30	0.036	0.007	389.6	61.522
	20	30 / 30	0.036	0.007	416.1	50.683
	21	30 / 30	0.034	0.027	299.3	31.378
Star	22	30 / 30	0.031	0.004	265.2	47.967
	23	30 / 30	0.027	0.007	161.0	27.261
	24	30 / 30	0.037	0.020	318.4	37.467

large value of R^2 suggests that there exists a strong linear relationship between density and solution time when taking into account the variance over the 30 replicates of each case. The number of iterations did not exhibit as strong a linear relationship ($R^2 = 0.348$) with network density for the networks instances tested.

6.3 Flow Model Extensions

Extensions to the baseline disrupting maximum flow model (*i.e.*, DMP-a) are proposed. The disrupting flow model is extended to consider limited resources, targeting utilizing mul-

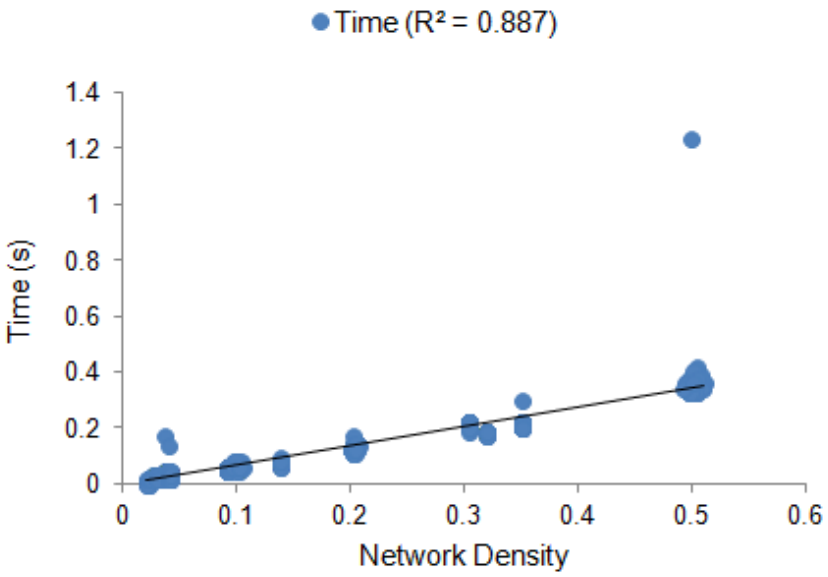


Figure 26. Density vs solution time for 100-node network instances tested

tiple strikes against a target, considering mission success at or above a threshold, and models to assist in visualizing the trade-offs between cost and residual flow. The model extensions are presented individually; however, the changes to transform a model from the baseline DMP-a model can be combined to formulate a more adaptable representation of the actual network disrupting scenario.

6.3.1 Disrupting with Limited Resources.

The DMP-a model is modified to account for a case where there are insufficient resources to target every arc contained in the cut set. To facilitate this change, the cut set is partitioned into those arcs selected and those not actually interdicted but still in the cut set similar to the method of Wood [68]. The variable z_{ij} is added to denote whether an arc (i, j) is in the s-t cut but not targeted. The interdiction resource cost required to disrupt the flow on an arc is denoted c_{ij}, and the total resource amount the attacker has available is R.

Since the leader's objective computes the minimum residual capacity cut, the follower's decision via the x-variable is not included. The network disrupting maximum flow problem for arcs only with resource constraints (**DMP-aR**) follows:

$$\min \quad \sum_{(i,j)\in E} y_{ij} b_{ij}(1 - r_{ij}) + \sum_{(i,j)\in E} (b_{ij} z_{ij}) - \varepsilon w \tag{6.3a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} c_{ij} y_{ij} + w = R, \tag{6.3b}$$

$$u_i - u_j + z_{ij} + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{6.3c}$$

$$u_s = 0, \tag{6.3d}$$

$$u_t = 1, \tag{6.3e}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{6.3f}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{6.3g}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{6.3h}$$

In this model, the objective function minimizes the total flow through the cut set via the first two terms in the objective. As the amount of interdiction resources diminish, the model is forced to allow flow across the cut set, and the first part of the objective determines this quantity. Constraints (6.3b), (6.3c), and (6.3g) ensure the desired partitioning of the s-t cut into those arcs interdicted and those that are not.

Constraint (6.3b) introduces a variable w representing the residual resource (equivalent to unused resource), and a multiple of this residual is included in the objective function to reward the conservation of resources (similar to Kallemyn *et al.* [44]). Because the residual is largest when conserving the resource, an ε multiple of the residual is added in the objective function. The ε should be small enough that the magnitude of the actual network flow is not altered, but not so small that the impact of resource conservation is lost. In effect, the

use of this term in the objective acts as a penalty function on the expenditure of interdiction resources and discriminates between any alternate optimal solutions with respect to the other objective function components.

The notional military transportation network described in Figure 25 and Table 53 is solved utilizing the DMP-aR model. Consider a resource availability of $R = 15$ units. The solution is to disrupt arcs (2,6), (3,6), (3,7), and (4,7) for a cost of 14 units of interdiction resource and allows a maximum flow across the residual network of 590 units when utilizing the capacity reductions indicated with $(r_{ij})_1$. When utilizing capacity reductions for $(r_{ij})_2$, the DMP-aR solution has a cost of 14 units, and the target set consists of arcs (7,10), (11,14), and (11,15). The maximum flow across the residual network is 426 units.

Thus, the decision maker is presented with a solution to disrupt the maximum flow on the residual network without exceeding the given resource limitation. The objective function is selected to provide the decision maker with the most disruptive target set while still staying within the resource budget.

6.3.2 Targeting for Multiple Strikes.

The destroy models in Chapter IV did not require a consideration for striking a targeted arc more than once since the models assume that an arc is completely destroyed upon attack. The disrupting models, however, relax this assumption and can consider multiple strikes against targeted arcs.

To account for multiple strikes against an arc within a maximum flow network, the DMP-aR model is modified. Notice that the baseline model partitions the cut set into those arcs that are targeted for a single strike and those that are not targeted at all. The partitioning of the cut set in the resource constrained model allows the allocation of multiple strikes while not targeting some arcs in the cut set whose capacity will remain unchanged.

The maximum number of strikes allowed for any targeted arc is denoted by ℓ. Thus, the decision variable y_{ij} takes an additional index that represents the number of strikes that are to be executed on arc (i, j). The index k can take integer values between 0 and ℓ, inclusively. The decision variable that indicates whether arc (i, j) is targeted for disruption is y_{ijk} where k represents the number of strikes against the arc. Notice that the variable z_{ij} is equivalent to y_{ij0}. Therefore, the variable y is used in lieu of z in the multiple strike model. The amount of resources consumed is computed by modifying Constraint (6.3b) to account for the additional allocation of resources for multiple strikes as

$$\sum_{k=1}^{\ell} \sum_{(i,j)\in E} k c_{ij} y_{ijk} + w = R.$$

The sum across the arcs targeted for multiple strikes have the respective amount of resource cost allocated via the multiplication by k.

Additionally, the model must account for the impact of reducing the arc's capacity by targeting an arc more than once. Thus, the objective term for minimizing the flow through the cut set becomes

$$\sum_{k=0}^{\ell} \sum_{(i,j)\in E} y_{ijk} b_{ij} (1 - r_{ij})^k.$$

This forces the model to account for the disruption of arcs due to multiple strikes against a single arc. The reduction in capability is raised to the power of the number of strikes. However, because the quantity k is a parameter and not a decision variable, the model remains linear. To ensure that an arc is selected to be targeted, regardless of the number of strikes, the sum of the y-variables across all strike possibilities is set less than or equal to 1, *i.e.*,

$$\sum_{k=0}^{\ell} y_{ijk} \leq 1, \quad \forall (i, j) \in E.$$

The follower's residual maximum flow problem is not included because the minimum residual capacity cut is utilized.

These changes allow the formulation of the network disrupting problem for arcs only with multiple strikes (**DMP-aM**):

$$\min \quad \sum_{k=0}^{\ell} \sum_{(i,j)\in E} y_{ijk} b_{ij} (1 - r_{ij})^k - \varepsilon w \tag{6.4a}$$

$$\text{s.t.} \quad \sum_{k=0}^{\ell} \sum_{(i,j)\in E} k c_{ij} y_{ijk} + w = R, \tag{6.4b}$$

$$u_i - u_j + \sum_{k=0}^{\ell} y_{ijk} \geq 0, \qquad \forall (i,j) \in E, \tag{6.4c}$$

$$u_s = 0, \tag{6.4d}$$

$$u_t = 1, \tag{6.4e}$$

$$\sum_{k=0}^{\ell} y_{ijk} \leq 1, \qquad \forall (i,j) \in E, \tag{6.4f}$$

$$y_{ijk} \in \{0,1\}, \qquad \forall (i,j) \in E, k = 0, \ldots, \ell \tag{6.4g}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{6.4h}$$

The notional military transportation network depicted in Figure 25 and Table 53 is utilized to demonstrate the DMP-aM model. The notional example is solved for the $(r_{ij})_1$ capacity reduction values with $R = 50$ and $\ell = 3$. The resulting disrupting solution contains arcs {(2,9), (5,9)} not targeted, arcs (2,6), (2,7), (3,6), (3,8), (4,8), (5,7)} targeted once each, and arcs {(3,7), (4,7), (5,8)} targeted for two strikes each for a total cost of 50 units and a residual flow of 392.5 units. When the interdiction budget is increased to $R = 100$, all available resources are consumed (*i.e.*, cost is 100 units) and residual flow decreases to 215.78 units by targeting arc (5,7) once, arcs {(2,6), (2,7), (3,6), (3,7), (3,8), (4,8), (5,8)} twice each, and striking arcs {(2,9), (4,7), (5,12)} three times each.

The example is also solved for the reductions in $(r_{ij})_2$ with $R = 50$ and $\ell = 3$. The DMP-aM model yields a solution that costs 50 units and allows 98.20 units of flow through the residual network. The cut set contains arcs {(5,12), (9,13), (9,14), (10,13), (10,14)}, selected to be targeted once each, and arcs {(8,12), (11,14), (11,15)} to be targeted twice each. When the resource allocation is increased to $R = 100$, the resulting solution targets arcs {(9,13), (9,14)} twice each and arcs {(5,12), (8,12), (10,13), (10,14), (11,14), (11,15)} three times each for a total cost of 99 unit and a residual flow of 11.38 units.

The decision maker is presented with a target allocation that maximally disrupts the residual network. A list of arcs that should be targeted and the number of strikes against each is presented. The expected damage is computed for the disruption based on the number of strikes planned against each arc.

6.3.3 Mission Success by Threshold.

For the disrupting flow model (*i.e.*, DMP-a), there may be a damage level as a result of an interdiction strategy that is large enough to consider the mission a success. This value could be considered a damage threshold value that, if known, would allow more flexible targeting options.

Utilizing the partition of the cut set into arcs targeted and not as in the limited resource model will allow the leader to save interdiction resource costs and ensure mission success by limiting flow to at most the threshold level. Let F denote the flow threshold for which the leader will consider the interdiction mission a success. The sum of the flow that traverses the cut set is utilized to ensure the flow does not exceed the threshold:

$$\sum_{(i,j)\in E} b_{ij}(1 - r_{ij})y_{ij} + \sum_{(i,j)\in E} b_{ij}z_{ij} \leq F.$$

The follower's residual maximum flow is not computed in the model so it is retained in the objective and constraints via the x-variable. The network disrupting maximum flow threshold problem for arcs only (**DMP-aT**).

$$\min \quad \sum_{(i,j)\in E} (c_{ij}y_{ij}) - \lambda x_{ts} \tag{6.5a}$$

$$\text{s.t.} \quad u_i - u_j + z_{ij} + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{6.5b}$$

$$u_s = 0, \tag{6.5c}$$

$$u_t = 1, \tag{6.5d}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = 0, \qquad \forall i \in V, \tag{6.5e}$$

$$x_{ij} \leq b_{ij}(1 - y_{ij}r_{ij}), \qquad \forall (i,j) \in E, \tag{6.5f}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{6.5g}$$

$$\sum_{(i,j)\in E} b_{ij}(1 - r_{ij})y_{ij} + \sum_{(i,j)\in E} b_{ij}z_{ij} \leq F, \tag{6.5h}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{6.5i}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{6.5j}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{6.5k}$$

The objective ensures that the leader selects a minimum cut set and that the follower maximizes flow on the residual network. Constraint (6.5h) ensures that the leader selects those arcs for interdiction such that the residual flow is at or below the threshold. The remaining constraints are as described for previous models.

Consider the notional military transportation example in Figure 25 with the capacity, cost and reduction levels $(r_{ij})_1$ from Table 53. If the flow threshold for mission success is $F = 680$, the solution for the DMP-aT model is to target arc (4,7) at a cost of 3 units with

a residual flow of 670 units. Alternatively, if the flow threshold is $F = 500$, the solution of DMP-aT model yields a target set of arcs {(2,9), (2,6), (2,7), (3,6), (3,7), (4,7), (5,7), (8,12)} for a cost of 32 units of interdiction resource and allows a maximum flow across the residual network of 497.5 units.

Thus, the decision maker receives a target solution that results in the residual maximum flow on the network below the desired threshold. The cost to disrupt the network is minimized. As the threshold becomes more aggressive (*i.e.*, the desired residual flow decreases), the number of arcs that should be targeted increases until the threshold value is satisfied.

6.3.4 Pareto Solutions.

The optimal solutions to the DMP-aR model are sensitive to changes in R. Decision makers that allocate resources to disrupting arcs in a network will be interested in trade-offs between arc interdiction costs and residual flow through the network. The Pareto solutions for the interdiction modeling may be found via an approach similar to the ϵ-constraint method of Ehrgott [24:pp. 98–101] or via an alternative explored by Royset and Wood [57]. For the disrupting problem, the DMP-aR model is extended to include features of the DTMP-a model to find solutions that enable the visualization of the Pareto solutions. The x-variable for the follower is not included since the objective solves the follower's maximum flow. The variable w is also not utilized since the Pareto solution will be strictly less than the previous cost solution.

The formulation of the network disrupting maximum flow problem for arcs only and Pareto solutions (**DMP-aP**) follows:

$$\min \quad \sum_{(i,j)\in E} y_{ij} b_{ij}(1 - r_{ij}) + \sum_{(i,j)\in E} z_{ij} b_{ij} \tag{6.6a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} c_{ij} y_{ij} \leq R - \varepsilon, \tag{6.6b}$$

$$\sum_{(i,j)\in E} y_{ij} b_{ij}(1 - r_{ij}) + \sum_{(i,j)\in E} z_{ij} b_{ij} \geq F + \varepsilon, \tag{6.6c}$$

$$u_i - u_j + z_{ij} + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{6.6d}$$

$$u_s = 0, \tag{6.6e}$$

$$u_t = 1, \tag{6.6f}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{6.6g}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{6.6h}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{6.6i}$$

The model identifies the minimum residual capacity cut (6.6a) subject to the total interdiction resource consumption less than or equal to the budget R minus a small $\varepsilon > 0$ such as 10^{-6} (6.6b), and the total flow greater than or equal to the threshold F plus a small $\varepsilon > 0$ (6.6c). The values of R and F are initially set to ∞ and 0, respectively, and the model is solved. The values of R and F are subsequently changed to the respective cost and flow solutions, and the model is solved again. This process is repeated until the solution has a total cost of 0 units. The cost and residual flow for each solution is plotted to generate a chart that allows the visualization of the Pareto solutions. The Pareto solutions may be modified to allow multiple strikes or any combination of the other extensions by incorporating the necessary modifications to the DMP-aP model.

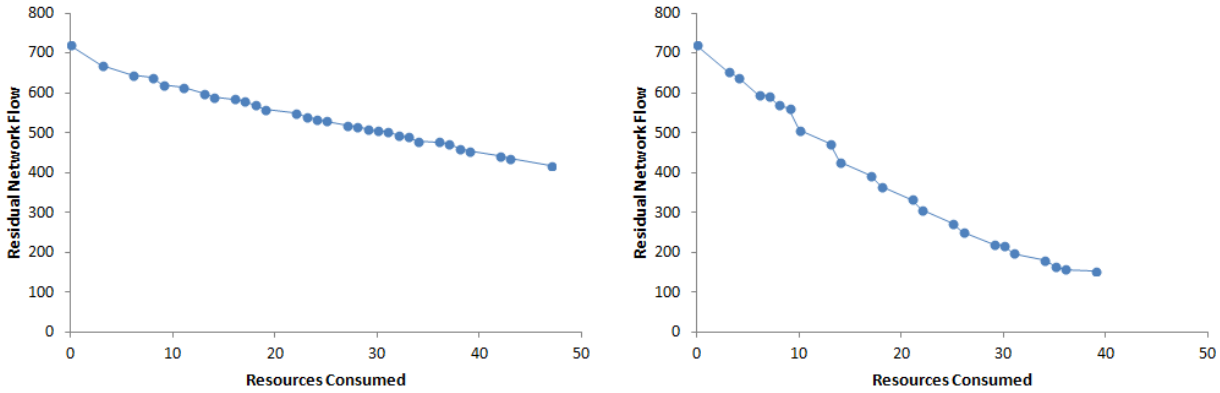


Figure 27. Pareto Solutions for $(r_{ij})_1$ (left) and $(r_{ij})_2$ (right) capacity reductions

Figure 27 depicts the Pareto solutions for the notional military transportation network utilizing the $(r_{ij})_1$ (left side) and $(r_{ij})_2$ (right side) reduction values. Notice that in each case, the flow in the residual network is not completely halted. The best the leader can do is apply the minimum capacity cut to disrupt as much of the follower's flow as possible.

Therefore, a decision maker can visualize the impact to the residual network as a result of changes in the resource budget. The more resources are allocated to the targeting problem, the smaller the residual flow is across the network. This trade-off visualization technique provides the decision maker with awareness to the trade-off between targeting cost and mission impact.

6.4 Summary

This chapter developed a number of models for network disrupting. This array of models allows a decision maker to determine the arcs that should be targeted to maximize the disruption of a network. These models represent attacks that do not completely destroy the targeted arc(s), but rather diminish its (their) capability. The decision maker can then ascertain the reduced capability of the adversary and plan additional missions to exploit the disruption.

The contributions are summarized in Table 56. The baseline model for shortest path disrupting was a modification of other work as an example. The maximum flow disrupting model allows the decision maker to understand the impact of partial interdiction. The maximum flow model was extended to consider limited resources, multiple strikes, a threshold for mission success, and a visualization of Pareto solutions. In the next chapter, network delaying tasks are considered.

Table 56. Disrupting Interdiction Task Contributions

Interdiction Task	Section	Contribution	Description
Disrupt	6.2	DSP-a	The disruptive shortest path problem for arcs only is similar to the shortest path interdiction models of Israeli and Wood [42].
	6.2	DMP-a	The disruptive maximum flow problem for arcs only utilizes features of DSP-a to formulate a new model for disrupting the maximum flow of a network.
	6.3.1	DMP-aR	Extends DMP-a model. The model represents disrupting maximum flow over a network with limited resources.
	6.3.2	DMP-aM	Extends DMP-a model. The model represents disrupting maximum flow over a network with targeting for multiple strikes.
	6.3.3	DMP-aT	Extends DMP-a model. The model represents disrupting maximum flow over a network with mission success if a threshold flow level is achieved.
	6.3.4	DMP-aP	Extends DMP-a model. The model represents identifying the Pareto solutions for disrupting maximum flow over a network.

VII. Delay Interdiction Tasks

7.1 Introduction

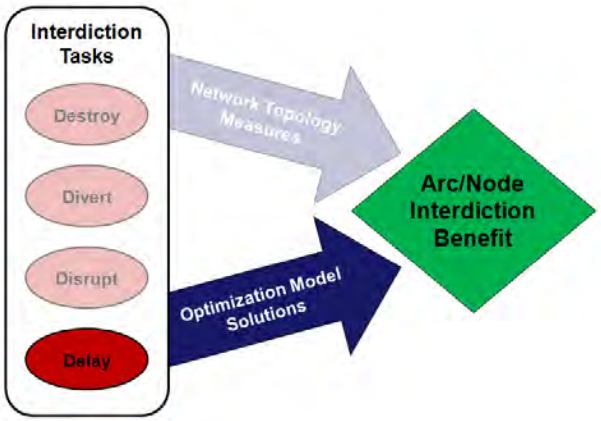


Figure 28. Research Framework: Modeling Delay Interdiction Tasks

In this chapter, a network interdiction modeling framework for delaying the restoration of network resources is developed. The focus in this chapter is the 'delay' interdiction task within the models approach of the research framework depicted in Figure 28. The chapter develops a suite of optimization models that selects arcs for destruction to maximize the time that the arcs will be unusable prior to their restoration.

Consider a military supply network. When an attacker plans a delay task, as defined in this research, any targeted arcs are destroyed. Subsequently, the network operator (*i.e.*, the military supply planners) must select alternate routes for their supplies or wait until reconstruction is complete so supplies again traverse the network. Meanwhile, the network operator will detect arc (*i.e.*, road or bridge) destructions and begin the process of simultaneously restoring all destroyed arcs to full capacity. The attacker utilizes the time taken by the operator to ascertain the situation, dispatch repair crews, and restore full functional-

ity so that other mission objectives may be executed while the adversary's capabilities are diminished.

Alternatively, consider a fiber optic communications network. Arcs may be utilized to represent the fiber optic cable connections between servers. As soon as a cable is cut, the network operator will ascertain which arc has been attacked. The network flow will correct for the missing arc, and the operator will immediately dispatch a repair crew to reconnect the affected servers with new fiber optic cable. The attacker may be able to conduct a denial of service attack on the diminished network that would have been ineffective had the arc not been severed. Conversely, the delay caused by the alternate routing may require enough time for some other operation to proceed undetected. In both cases, it is assumed that the network operator can immediately detect the destruction of an arc and has sufficient repair resources to simultaneously repair any number of destroyed arcs.

A number of network interdiction models in the literature assume that arcs or nodes are completely destroyed or damaged and the residual capabilities of the network remain at that damaged level for the remainder of the model time horizon. However, an adversary will begin to restore his or her capabilities as soon as it is practical to do so. When attacking an adversary, it is more realistic to account for the time that the enemy capability is diminished and monitor the progress of restoration. A minimum restoration time objective is developed for a maximum flow network interdiction problem. Solutions for the minimum restoration time objective are equivalent to solutions for which the objective is maximizing the delay in restoring the network's capabilities.

7.2 Model Development

Given a network with sets of nodes V and arcs E along with a source node s and terminus node t, a typical minimum cut formulation (for arcs) [68] follows:

$$\min \quad \sum_{(i,j)\in E} b_{ij} y_{ij} \tag{7.1a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{7.1b}$$

$$u_s = 0, \tag{7.1c}$$

$$u_t = 1, \tag{7.1d}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{7.1e}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{7.1f}$$

In this formulation the binary decision variable y_{ij} indicates whether arc (i,j) is interdicted ($y_{ij} = 1$) or not ($y_{ij} = 0$). The binary variable u_i denotes whether node i is on the source side of the cut ($u_i = 0$) or the terminus side ($u_i = 1$). The objective (7.1a) ensures that the minimum capacity cut-set is selected with b_{ij} representing the capacity of arc (i,j). Constraint (7.1b) ensures that a cut is identified. The source and terminus nodes are indicated via (7.1c) and (7.1d), respectively.

For many network interdiction problem models, two adversaries observe the same network, and the leader (interdictor) attempts to cut the source-terminus flow, after which the follower (network operator) seeks to maximize the flow on the residual network. The minimum cut formulation ensures that all flow is cut, so the leader's decision is determined explicitly, with the implication is that the follower is left with a network that has no source-terminus flow.

The minimum cut formulation as given in (7.1) assumes that the interdicted arcs remain interdicted (completely destroyed) for the duration of the leader's mission. However, the

interdictor may need to account for the time that the interdiction will be effective [56]. In other words, the attacker may want to account for the restoration time of the network operator to ensure the interdiction is effective for the desired duration for the ensuing mission. Therefore, assume that an arc is in one of three states: it is not targeted and allows flow up to its capacity, it has been attacked and allows no flow because it has not yet been restored, or it has been attacked but allows flow up to its capacity after the allocated restoration time. Note that the models developed in this research assume complete restoration when the time to restore an arc is realized. A more realistic partial restoration assumption may be implemented as an extension of the baseline model.

Rocco *et al.* [56] propose this objective as a possibility when proposing multiple objectives for the network interdiction problem. They do not present it in a mathematical programming model but rather utilize Monte Carlo simulation to predict good solutions to multi-objective network interdiction problems.

Figure 29 depicts a scenario for arcs being restored (all arcs are assumed to be restored simultaneously) and the maximum flow increasing as arcs again become usable. At time 0 the interdiction task is accomplished and the network has a maximum flow indicated by the shaded area during the first time step. All flow on the network may not be eliminated as depicted in the figure when arcs are not impacted by the interdiction task, *i.e.*, the restoration

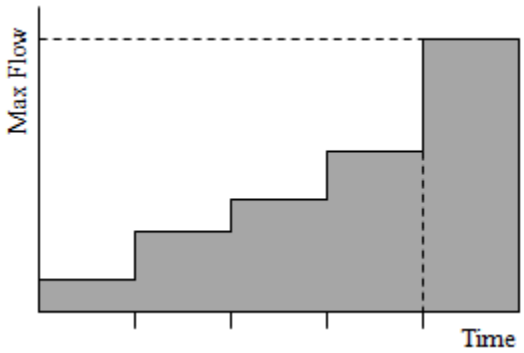


Figure 29. Example network response from interdiction [56]

time has value 0. A number of arcs will be restored in the first time period. At that time, the targeted arcs are restored, and the maximum flow adjusts to the maximum level for the current residual network. At the completion of the next longest restoration time, additional arcs regain their full capacity and a new maximum flow is realized for the residual network. Eventually, all arcs in the cut set are assumed to be restored and the maximum network flow is realized (right of the vertical dashed line representing the end of the mission's duration). Every targeted arc will eventually be restored since the maximum restoration time is given explicitly. Uniform time increments for arc restoration are not required as depicted in the figure.

Consider a finite time horizon over which the leader seeks to minimize the follower's maximum network flow. Denote this mission duration time by T. In the context of network diverting, the mission duration is the amount of time the interdictor desires that the divert set is rendered unusable for the network operator. Let $t_{ij} \in \{0, 1, 2, \ldots, T\}$ denote the time to restore arc (i, j), where a 0 time to restore implies that the arc will operate up to full capacity whether or not it is targeted for interdiction. The shaded area to the left of the dashed line in Figure 29 represents the maximum flow over the mission duration. This quantity represents a proxy for the speed with which the network is restored. Thus, an objective that minimizes the area effectively will minimize the potential restoration time or, equivalently, will maximize the delay in restoration of the network.

The area of the maximum flow over time is given by

$$(T-0) \sum_{\substack{(i,j)\in A \\ y_{ij}=1 \\ t_{ij}=0}} b_{ij} + (T-1) \sum_{\substack{(i,j)\in A \\ y_{ij}=1 \\ t_{ij}=1}} b_{ij} + \cdots + \sum_{\substack{(i,j)\in A \\ y_{ij}=1 \\ t_{ij}=T-1}} b_{ij}. \tag{7.2}$$

This expression computes the total of the maximum flow over each time increment up to T. Each term effectively accounts for the increase in capacity of the maximum flow for each time

interval in the mission duration. Notice that the variable y_{ij} is binary; each sum contains only a specific set of the arcs based on the value of t, and the entire sum is multiplied by the difference between the mission duration and the restoration time. Moving this difference inside each summation, substituting the value of t, and accounting for the binary state of the variable yields

$$\sum_{(i,j)\in A} b_{ij} y_{ij} (T - t_{ij}). \tag{7.3}$$

This expression accounts for the contribution of any interdicted arcs to the maximum allowable flow after their restoration time is met. The indicator variable y_{ij} ensures that only arcs in the cut are calculated, and limits the restoration times to be at most T allows the calculation of the maximum flow over time to be used as a proxy for the minimum total restoration time from the leader's perspective.

The formulation of the minimum restoration time problem for arcs (**MRTP-a**) follows:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} b_{ij} y_{ij} (T - t_{ij}) && && \text{(7.4a)}\\
\text{s.t.} \quad & u_i - u_j + y_{ij} \geq 0, && \forall (i,j) \in E, && \text{(7.4b)}\\
& u_s = 0, && && \text{(7.4c)}\\
& u_t = 1, && && \text{(7.4d)}\\
& y_{ij} \in \{0,1\}, && \forall (i,j) \in E, && \text{(7.4e)}\\
& u_i \in \{0,1\}, && \forall i \in V, && \text{(7.4f)}
\end{aligned}$$

with the objective as described for (7.3) and the constraints unchanged from the minimum cut formulation (7.1b)-(7.1f).

This model may be appropriate if the attacker knows the duration of the restoration for arcs but does not know the prioritization for the order in which arcs are restored or how many crews will be assigned to work to simultaneously restore arcs. In other words, this model presents a worst-case (for the attacker) restoration effect that is accounted for by the leader planning the interdiction so as to minimize the amount of time that targeted arcs operate at full capacity.

7.3 Testing

The MRTP-a model is tested on network instances that are generated as described in Appendix A. The parameters utilized for each network case are given in Table 57. For each of the 24 network structure cases listed, 30 replicates were generated. Additionally, a reference for each network type is listed denoting the particular parameter setting utilized as well as the average density for the network type over the 30 replicates. The values of the restoration time-weight and the capacity-weight assigned to each arc in the network are selected using a discrete (integer) uniform distribution having a range between 1 and 10, inclusively. The value of the mission duration is 10, *i.e.*, $T = 10$, in accordance with the utilization of restoration times in the model.

The MRTP-a model is solved for the various random network instances using IBM® ILOG® CPLEX® Optimization Studio V12.6. For each instance, two solutions are determined. First, the MRTP-a model is solved with the y-variable relaxed to a continuous variable bounded by 0 and 1. Second, the MRTP-a model is solved as stated in the model formulation with the y-variables binary. In both cases, the u-variables are binary. The cases test whether the solver is more suited for the given model utilizing the decision variable y as binary or continuous.

Table 58 displays the MRTP-a results for the 100-node network instances. The network structure and case reference are listed in the first two columns. The case values are found

in Table 57. For each of the solution approaches (*i.e.*, y-variables relaxed or restricted to binary values), the number of instances solved to optimality, mean solution time, and number of iterations required are recorded. Finally, for the replicates of each network type, the two-tailed paired-t test statistic is utilized to determine whether there is a difference in the measure (*i.e.*, mean solution time or number of iterations) with y-variables relaxed or binary. In cases when the p-value is less than 0.05, there is sufficient evidence at the 0.05 level of significance to reject the claim that there is no difference in the measure when the y-variables are relaxed or binary. For larger p-values, there is insufficient evidence at the 0.05

Table 57. Parameter settings for 100-node random network structures

Structure	Parameters	Case	Mean Density
ER	$\beta = 0, p = 0.1$	1	0.098
	$\beta = 1, p = 0.1$	2	0.099
	$\beta = 0, p = 0.5$	3	0.500
	$\beta = 1, p = 0.5$	4	0.502
BA	$m_a = 2, n_0 = 25$	5	0.091
	$m_a = 5, n_0 = 25$	6	0.136
	$m_a = 2, n_0 = 55$	7	0.318
	$m_a = 10, n_0 = 50$	8	0.348
WS	$k = 4, p = 0.10$	9	0.040
	$k = 4, p = 0.25$	10	0.040
	$k = 30, p = 0.10$	11	0.303
	$k = 30, p = 0.25$	12	0.303
PNDCG	$\beta = 0, \alpha = 2.35$	13	0.025
	$\beta = 1, \alpha = 2.35$	14	0.025
	$\beta = 0, \text{dist} = U_{3,5}$	15	0.030
	$\beta = 1, \text{dist} = U_{3,5}$	16	0.030
	$\beta = 0, \text{dist} = U_{20,5}$	17	0.202
	$\beta = 1, \text{dist} = U_{20,5}$	18	0.202
Grid	10×10	19	0.037
	5×20	20	0.038
	20×5	21	0.035
Star	10×10	22	0.040
	5×20	23	0.040
	20×5	24	0.040

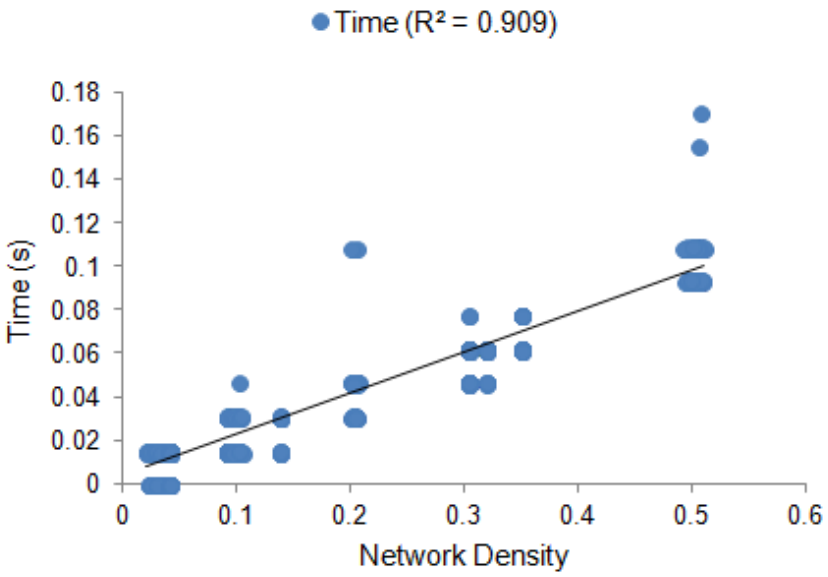


Figure 30. Density vs solution time for 100-node network instances tested

level of significance to reject the claim of no difference in the measure when the y-variables are relaxed or binary.

The p-values corresponding to mean solution time are larger than 0.05 for all network types except the low-density networks labeled Case 2 (ER) and 7 (BA). In addition, the p-values for the number of iterations are larger than 0.05 for network Cases 5, 8, and 15. In every network type having sufficient evidence at the 0.05 level of significance to reject the claim that no difference in the measure, the binary model instance yielded an optimal solution either more quickly or in fewer iterations. Therefore, for the network instances tested, the model should be solved utilizing binary decision variables as denoted in the model statement, all other things being equal.

Network density is graphed against solution times in Figure 30 for the network instances tested. The line represents a linear approximation of the relationship between density and solution time. For the networks tested, solution time increases with an associated increase in network density. The coefficient of determination R^2 between density and solution time,

given in parentheses in the legend of the chart, for the tested network cases is 0.909. The large value of R^2 suggests that there exists a strong linear relationship between density and solution time when taking into account the variance over the 30 replicates of each case. The number of iterations did not exhibit a linear relationship ($R^2 = 0.001$) with network density for the networks instances tested.

Table 58. Comparison of results for MRTP-a with y relaxed and binary

Structure	Case	y Relaxed			y Binary			2-Tail p-value	
		# Solved	Mean Time	Mean Its	# Solved	Mean Time	Mean Its	Time	Iterations
ER	1	30 / 30	0.022	97.2	30 / 30	0.023	97.0	0.706	0.096
	2	30 / 30	0.019	107.2	30 / 30	0.023	106.3	0.042	0.167
	3	30 / 30	0.100	106.9	30 / 30	0.106	106.8	0.106	0.161
	4	30 / 30	0.104	111.1	30 / 30	0.117	111.1	0.186	1
BA	5	30 / 30	0.019	3.6	30 / 30	0.017	3.3	0.152	0.039
	6	30 / 30	0.027	10.2	30 / 30	0.028	10.2	0.782	1
	7	30 / 30	0.059	3.1	30 / 30	0.050	3.0	0.006	0.161
	8	30 / 30	0.068	12.9	30 / 30	0.065	12.8	0.053	0.043
WS	9	30 / 30	0.013	102.4	30 / 30	0.012	102.1	0.540	0.067
	10	30 / 30	0.013	99.7	30 / 30	0.014	99.6	0.412	0.662
	11	30 / 30	0.059	129.0	30 / 30	0.061	128.9	0.564	0.161
	12	30 / 30	0.058	117.2	30 / 30	0.059	117.1	0.659	0.161
PNDCG	13	30 / 30	0.013	57.2	30 / 30	0.014	53.2	0.402	0.098
	14	30 / 30	0.011	52.5	30 / 30	0.011	51.3	0.963	0.353
	15	30 / 30	0.009	80.1	30 / 30	0.011	79.2	0.278	0.026
	16	30 / 30	0.010	72.1	30 / 30	0.012	71.2	0.247	0.076
	17	30 / 30	0.041	108.8	30 / 30	0.043	108.7	0.196	0.326
	18	30 / 30	0.042	108.2	30 / 30	0.041	108.1	0.743	0.103
Grid	19	30 / 30	0.012	130.9	30 / 30	0.012	130.3	0.807	0.260
	20	30 / 30	0.013	121.3	30 / 30	0.014	120.9	0.345	0.317
	21	30 / 30	0.011	117.6	30 / 30	0.013	117.9	0.276	0.442
Star	22	30 / 30	0.014	113.5	30 / 30	0.014	113.2	0.711	0.161
	23	30 / 30	0.011	61.4	30 / 30	0.011	60.7	0.782	0.230
	24	30 / 30	0.012	126.8	30 / 30	0.011	126.8	0.820	1

7.4 Model Extensions

The models that maximize the delay of restoration can be modified to account for several extensions. Each of the modifications are implemented individually, unless noted otherwise. Multiple changes can be incorporated in a single model if necessary by making each of the appropriate modifications.

The MRTP-a model is modified utilizing the nodal interdiction method described by Kennedy *et al.* [45]. The decision variable v_i is introduced where $v_i = 1$ if node i is interdicted and $v_i = 0$ otherwise. With the modification to allow nodal interdiction, restoration time t_i for each node is introduced. This leads to a formulation for the minimum restoration time problem for nodes only (**MRTP-n**):

$$\min \quad \sum_{(i,j)\in E} b_{ij} y_{ij} (T - t_i) \tag{7.5a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{7.5b}$$

$$u_s = 0, \tag{7.5c}$$

$$u_t = 1, \tag{7.5d}$$

$$y_{ij} = v_i, \qquad \forall (i,j) \in E, \tag{7.5e}$$

$$v_\ell = 0, \qquad \ell = s, \ell = t, \tag{7.5f}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{7.5g}$$

$$v_\ell \in \{0,1\}, \qquad \forall \ell \in V, \tag{7.5h}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{7.5i}$$

In this model, the objective maximizes the delay of restoring nodes in the residual network. The arc capacities of outbound arcs from the interdicted node are restored after the node's restoration time is realized. The addition of Constraint (7.5e) enforces the cut of outgoing

arcs from the interdicted node. Constraint (7.5f) ensures that the source and terminus are not interdicted. The decision variable indicating which nodes are interdicted is binary in Constraint (7.5h). All remaining constraints are as described for the MNDP-a model.

To modify the MRTP-n to allow both node and arc interdiction, the objective is modified to account for node and arc costs. The model interdicts every outgoing arc from a node when it is attacked via the y-variable. Therefore, it is necessary to separate the arc and node restoration activities. Additionally, Constraint (7.5e) becomes $y_{ij} \geq v_i, \forall (i,j) \in E$. This allows for either arc or node interdiction. Thus, the formulation of the minimum restoration time problem for both arcs and nodes (**MNDP-b**) is:

$$\min \quad \sum_{(i,j)\in E} \left(b_{ij}(y_{ij} - v_i)\right)(T - t_{ij}) + \sum_{(i,j)\in E} b_{ij} y_{ij}(T - t_i) \tag{7.6a}$$

$$\text{s.t.} \quad u_i - u_j + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{7.6b}$$

$$u_s = 0, \tag{7.6c}$$

$$u_t = 1, \tag{7.6d}$$

$$y_{ij} \geq v_i, \qquad \forall (i,j) \in E, \tag{7.6e}$$

$$v_\ell = 0, \qquad \ell = s, \ell = t, \tag{7.6f}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E, \tag{7.6g}$$

$$v_\ell \in \{0,1\}, \qquad \forall \ell \in V, \tag{7.6h}$$

$$u_i \in \{0,1\}, \qquad \forall i \in V. \tag{7.6i}$$

The objective (7.6a) separates the arc (first summand) and node (second summand) contributions to the allowable flow after the respective arc or node restoration time is satisfied. The remaining constraints are as described for the MNDP-n model with the modification of the inequality for Constraint (7.6e).

Finally, the MRTP-a formulation is modified to account for a case where there are insufficient resources to interdict each arc in the cut set. To facilitate this change, a partition of the cut into those selected and those not actually interdicted but still in the cut set is utilized, similar to the method of Wood [68]. The variable z_{ij} is added which denotes whether arc (i, j) is in the cut but not targeted for attack. The amount of interdiction resources required to target the flow on an arc is denoted r_{ij} and the amount the attacker has available is R. This gives the minimum restoration time problem for arcs only with resource constraints (**MRTP-aR**).

$$\min \quad \sum_{(i,j)\in E} b_{ij}y_{ij}(T - t_{ij}) + b_{ij}z_{ij}T - \varepsilon w \tag{7.7a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} r_{ij}y_{ij} + w = R, \qquad \forall (i,j) \in E, \tag{7.7b}$$

$$u_i - u_j + z_{ij} + y_{ij} \geq 0, \qquad \forall (i,j) \in E, \tag{7.7c}$$

$$u_s = 0, \tag{7.7d}$$

$$u_t = 1, \tag{7.7e}$$

$$y_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in E, \tag{7.7f}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in E, \tag{7.7g}$$

$$u_i \in \{0, 1\}, \qquad \forall i \in V. \tag{7.7h}$$

In this model, the objective is partitioned in accordance with the partitioning of the cut set. The first term accounts for the contribution of any interdicted arcs to the maximum flow after its restoration time is met. The second term accounts for the contribution of any arcs in the cut set but not targeted to the maximum flow throughout the mission duration. The indicator variables y_{ij} and z_{ij} ensure that only arcs in the cut are calculated, yielding the

average maximum flow time which is used as a proxy for the minimum restoration time from the attacker's perspective.

Constraint (7.7c) ensures that the available interdiction resource allotment is not exceeded and that only those arcs selected to be in the cut and targeted are counted toward the resource consumption. Constraint (7.7c) ensures that the cut set is partitioned into those arcs that are targeted via the y variable and those that are not via the z variable. The decision variable indicating the arcs in the cut that are not targeted is binary in Constraint (7.7g).

As the amount of interdiction resources diminish, the model selects arcs for the cut that will operate at full capacity for the entire time horizon. Constraint (7.7b) introduces a slack-like variable w representing the residual resource (equivalent to unused resource) and a multiple of this residual is included in the objective function to reward the conservation of resources (similar to the construct of Kallemyn *et al.* [44]). Because the residual is largest when conserving the resource, an ε multiple of the residual resource is added in the minimizing objective. The ε should be small enough that the magnitude of the actual network flow is not altered, but not so small that the effect of resource conservation is lost. In effect, the use of this term in the objective acts as a penalty function on the expenditure of interdiction resources and discriminates multiple optimal solutions, if any exist.

7.5 Summary

This chapter described and developed models to delay the restoration of a network after interdiction. The suite of models developed in this chapter allows a decision maker to determine the arcs that should be targeted to maximize the delay in the network's restoration. This is accomplished by maximizing the estimated down time for the arcs. With this knowledge, a decision maker can plan operations that exploit the delay and execute missions undetected.

The contributions of this chapter are summarized in Table 59. A minimum cut formulation was modified to model maximizing the delay in network restoration. This model was solved for several test instances. Testing demonstrated that utilizing binary decision variables in the solver was appropriate and the density of the network topology impacts solution times for the tested network instances. Finally, testing illustrated that on network instances having the same density and structure but different parameter settings, solution times may not be impacted, but the number of iterations might.

Table 59. Delaying Interdiction Task Contributions

Interdiction Task	Section	Contribution	Description
Delay	7.2	MRTP-a	The minimum restoration time problem for arcs only accounts for the contribution of any interdicted arcs to the maximum allowable flow after their restoration time is met.
	7.4	MRTP-n	Extends MRTP-a model. The model represents delaying the restoration of nodes contained in a network based on maximum flow.
	7.4	MRTP-b	Extends MRTP-a model. The model represents delaying the restoration of arcs and/or nodes contained in a network.
	7.4	MRTP-aR	Extends MRTP-a model. The model represents delaying the restoration of arcs contained in a network with limited resources.

VIII. Conclusion

8.1 Summary

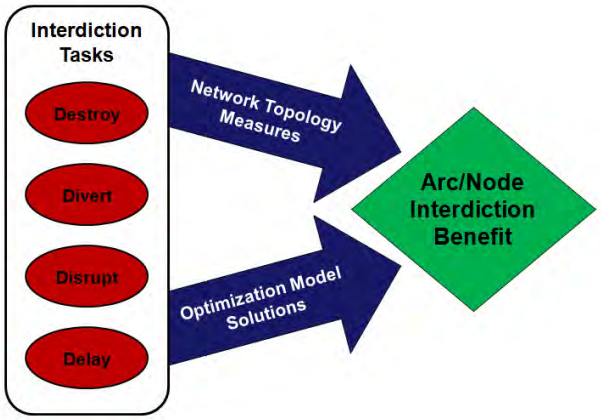


Figure 31. Research Framework

The research framework is again depicted in Figure 31. Each of the chapters in the dissertation addressed determining the interdiction benefit of a node or arc from a measures or models perspective. The four interdiction tasks outlined in joint defense doctrine provided the framework from which to represent them in the models employed. The measures and models comprise a suite of tools a decision maker can utilize to determine the best targets for a given scenario and interdiction task combination.

8.2 Contributions

A number of theoretical developments and extensions or applications were developed and tested in this dissertation. These contributions are summarized in this section. Each approach (*i.e.*, measures and models) is discussed separately.

In the measures approach of the dissertation, three contributions were developed. The measures approach focused on the 'destroy' interdiction task in Chapter III. First, an algo-

rithm used in practice was extended to account for all geodesics and a number of measures. This contribution is an extension and application of previous work. Next, a new approach was developed to account for the removal of each node contained in a network. This theoretical contribution utilizes the extended algorithm to determine the impact each node's removal has on geodesic lengths within the residual network. Finally, a number of new measures were introduced to assess the impact of the removal of nodes in a network.

Whereas the measures approach considered the 'destroy' interdiction task only, the models approach examined each of the four interdiction tasks identified in joint doctrine, *i.e.*, 'destroy', 'divert', 'disrupt', and 'delay' tasks. For the models approach to 'destroy' interdiction tasks in Chapter IV, the theoretical contribution includes the development of a new modeling approach for identifying the arcs in the network whose destruction result in interdicting the k shortest paths.

The 'divert' interdiction task was previously undefined in the open literature as defined in joint doctrine. To address this theoretical gap in modeling interdiction tasks, a modeling approach for diverting as defined in joint doctrine was developed in Chapter V. To accomplish this development, mathematical programming techniques common to network interdiction and multicriteria optimization were applied to solve the newly posed diverting problem.

The models approach for 'disrupt' interdiction tasks extended the approach of others, which increased length of shortest paths, in a new application for modeling disruptions in maximum flow network models. Additionally, a new problem was proposed to determine threshold for expected damage to ensure successful interdiction.

Finally, in Chapter VII the 'delay' interdiction task was examined. An ill-defined objective function utilized in other work for Monte Carlo simulation was transformed to a network interdiction model objective function. This new objective function allowed the delay in network restoration time to be maximized in maximum flow models.

These developments are listed with a section reference and brief description in Table 60.

Table 60. Dissertation Contributions

Interdiction Task	Section	Contribution	Description
Destroy	3.3.1	EAGL Algorithm	The Extended All Geodesics Lengths (EAGL) Algorithm takes the features of previous algorithms (*i.e.*, retaining all geodesic and predecessor information, computing node dependencies, and counting occurrences of nodes on geodesics) to construct a more robust algorithm.
	3.4.1	GAND Approach	The Geodesics After Node Destruction (GAND) Approach determines the impact on the geodesic lengths between all node pairs in the network as a result of each node's destruction.
	4.2	MAD	The maximum flow arc destroying model ensures that the leader's minimum cost source-terminus cut set is selected eliminating the follower's flow.
	4.3	SAD	The shortest path arc destroying model identifies a single arc in the intersection of the leader's source-terminus cut set and the follower's shortest path.
	4.3	SAD-kI	Extends SAD model. The model selects k arcs in the leader's minimum cost cut set to interdict the k shortest arc-independent paths.
	4.3	SAD-k	Extends SAD model. The model identifies the k arcs in the leader's minimum cut set that interdict the follower's k shortest paths.

Continued on next page

Table 60 – *Continued from previous page*

Interdiction Task	Section	Contribution	Description
Divert	5.2	MNDP-a	The maximum flow network diverting problem for arcs only determines the leader's minimum cost source-to-divert set cut while maximizing the remaining source-to-terminus flow on the network for the follower.
	5.3	SNDP-a	The shortest path network diverting problem for arcs only determines the leader's minimum cost source-to-divert set cut to ensure no path for the follower traverses the divert set.
	5.4	SNDP-aM	Extends SNDP-a model. The model represents diverting shortest paths from multiple divert sets.
	5.4	MNDP-n	Extends MNDP-a model. The model represents diverting the follower's residual flow from the divert set by targeting nodes.
	5.4	MNDP-b	Extends MNDP-a model. The model represents diverting the follower's residual flow from the divert set by targeting arcs and/or nodes.
	5.4	SNDP-aT	Extends SNDP-a model. The model represents targeting arcs such that the total distance traversed within the divert set is less than a threshold.

Continued on next page

Table 60 – *Continued from previous page*

Interdiction Task	Section	Contribution	Description
	5.4	MNDP-aR	Extends MNDP-a model. The model represents diverting the residual flow from the divert set with limited resources.
Disrupt	6.2	DSP-a	The disruptive shortest path problem for arcs only is similar to the shortest path interdiction models of Israeli and Wood [42].
	6.2	DMP-a	The disruptive maximum flow problem for arcs only utilizes features of DSP-a to formulate a new model for disrupting the maximum flow of a network.
	6.3.1	DMP-aR	Extends DMP-a model. The model represents disrupting maximum flow over a network with limited resources.
	6.3.2	DMP-aM	Extends DMP-a model. The model represents disrupting maximum flow over a network with targeting for multiple strikes.
	6.3.3	DMP-aT	Extends DMP-a model. The model represents disrupting maximum flow over a network with mission success if a threshold flow level is achieved.
	6.3.4	DMP-aP	Extends DMP-a model. The model represents identifying the Pareto solutions for disrupting maximum flow over a network.

Continued on next page

Table 60 – *Continued from previous page*

Interdiction Task	Section	Contribution	Description
Delay	7.2	MRTP-a	The minimum restoration time problem for arcs only accounts for the contribution of any interdicted arcs to the maximum allowable flow after their restoration time is met.
	7.4	MRTP-ac	Extends MRTP-a model. The model represents delaying the restoration of nodes contained in a network based oon node capacities.
	7.4	MRTP-n	Extends MRTP-a model. The model represents delaying the restoration of nodes contained in a network based on maximum flow.
	7.4	MRTP-b	Extends MRTP-a model. The model represents delaying the restoration of arcs and/or nodes contained in a network.
	7.4	MRTP-aR	Extends MRTP-a model. The model represents delaying the restoration of arcs contained in a network with limited resources.

8.3 Future Research

In Chapter III, there appeared to be a density threshold at which the F-W algorithm becomes more efficient than a repeated application of the BFS or Dijkstra's algorithm (possibly when density is near 0.35). However, in the open literature, this threshold has not been explored. "There is no strict distinction between sparse and dense [networks]." [8] General guidance indicates that the F-W algorithm is appropriate for dense graphs and RD-type algorithms are more suited to sparse graphs. A designed experiment and analysis of computation times for the geodesic matrix utilizing the two methods would provide an initial estimate for the value of such a threshold.

As outlined in Chapter III, when studying the impact of large-scale node removals to networks, it is important to quickly characterize the connectedness of the residual network and the size of the largest connected component. These features are computed utilizing the eigenvalues and associated eigenvectors of the Laplacian matrix. This analysis may be completed and will be a useful addition to the suite of interdiction models and measures available to decision makers.

The network diverting models developed in this research utilize minimum cost cuts to model the leader problem. An alternative is to utilize the isolation set model to determine the nodes and/or arcs that should be targeted to separate the divert set from the rest of the network.

The models currently utilized to assess interdiction operations against a network do not consider multiple interdiction tasks being completed simultaneously. It is likely that mission objectives will require that different interdiction tasks be accomplished in nearby areas of the network. The compounding and cascading effects of interdiction activities can be combined to accomplish the goals more efficiently in a single model.

Furthermore, the majority of interdiction models in the literature generally assume interdiction tasks are directed against a single-layered network. While it is possible that an adversary's infrastructure can be modeled as a single-layer network with interconnections, perhaps it nay be more appropriate to model interdiction tasks against multi-layered networks. Separating individual layers within the infrastructure into functional networks with interdependencies connecting the layers will allow analysis of the interdiction benefit through decomposition methods. This allows the identification of critical nodes and/or arcs in an adversary's multi-layered infrastructure, given a specific interdiction task or combination of interdiction tasks.

Finally, an interdiction task in an infrastructure network can be indirect. An indirect interdiction task would affect nodes or arcs in other areas of a network but with the desired goal taking effect in a specific area of the network. By considering interdiction activities exclusively outside the area in which the goal is to take effect, the mission is accomplished without bringing heightened awareness to the objective. This type of analysis will assume that the cascading and compounding effects are intentional and that an attack in one area is used to realize a deliberate effect in another area of the network.

8.4 Concluding Remarks

This research developed a suite of models and measures to aid decision makers in deliberate and rapid planning and analysis of interdiction tasks. The interdiction benefit of nodes and/or arcs was determined for destroying, diverting, disrupting, and delaying network capabilities. This array of measures and models may serve as modeling options for offensive and defensive operations. The operations that can be considered when utilizing this suite of measures and models may be either kinetic or non-kinetic actions such as detailed observation, signals collection, denial of service, or possibly destruction. Thus, the suite of models and measures developed in this dissertation provide a foundation for analysis of operational offensive and/or defensive plans and a launch point for future research.

Appendix A. Test Plan

The measure algorithms and optimization models in this research were tested on notional networks. The level of testing performed varied depending on the nature of the performance claim. Jackson *et al.* [43] identified three potential performance claims and the appropriate level of testing to support the claim. First, preliminary testing "on several well-chosen problems would probably suffice" for demonstrating the feasibility of the implementation of an algorithm or the models from which numerical results are produced [43:p.416]. The next level of testing is more detailed; the "strengths and weaknesses of an implementation" can be assessed using an appropriately chosen range of problems [43:p.416]. Finally, a "detailed comparison of the performance... with prominent methods already available" should be used to substantiate performance claims. The measures assessed should be comparable for each of the methods tested [43:p.416]. These guides were used where appropriate to substantiate performance claims when a computation was made in this research.

Random networks are useful because a number of networks can be generated and stored for testing by replicating the random draws that define weights within the network. Several instances of random networks are considered. Grid and star-mesh networks have a fixed topology for given size parameters. Random networks are generated to have features that mimic types of real-world networks. Finally, random networks can be constructed using a network generator.

Grid networks are the most straight-forward to implement and have been used to test many of the network interdiction models in the open literature. A grid network is comprised of nodes aligned on an $h \times v$ grid that are connected to other nodes along vertical and horizontal arcs within the grid. Figure 32 illustrates a 4×4 grid network with source s and terminus t.

Star-mesh networks are also relatively straight-forward. A central node is the source and has c concentric rings each with r nodes equally spaced around the ring along rays

emanating from the central node. The nodes are connected along the rings and along the spokes. A terminus node is selected from the nodes on the outermost ring. A notional star-mesh network having $c = 2$ concentric rings, each containing $r = 8$ nodes along rays emanating from the center, is depicted in Figure 33.

Random grid and star-mesh networks have been used to test network interdiction models. A number of papers related to network diversion tested interdiction models using these network types [19] [20] [26]. In addition, it may be useful to consider random networks. Erdős-Rényi networks, Barabási-Albert networks, and Watts-Strogatz networks are typically used to generate random networks.

Erdős-Rényi (ER) networks are constructed by connecting a set of all nodes randomly [25]. The parameter p represents the probability that any given node-pair is connected and there-

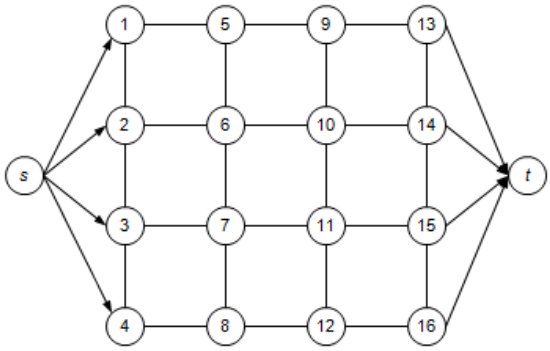


Figure 32. Example of a grid network

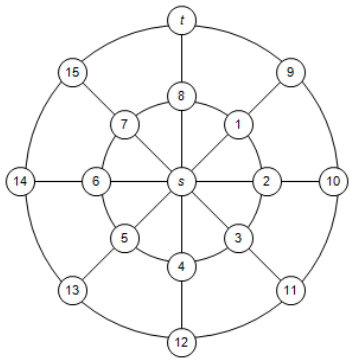


Figure 33. Example of a star-mesh network

fore is a proxy for the density. Figure 34 depicts an ER network where each of 10 nodes is connected with a probability p of 25 percent. Because the arcs are included in the network with probability p, the degree distribution for any node is binomial [54].

Barabási-Albert (BA) networks are constructed by completely connecting a set of initial nodes and connecting additional nodes (preferring the most connected nodes) until the desired total is reached [4]. The parameter n_0 denotes the number of nodes that are initially completely connected and m_a denotes the number arcs that are connected for each additional node until the desired total is attained. A BA network is illustrated in Figure 35 that was constructed utilizing an initial set of $n_0 = 6$ nodes and connecting a single node via a single arc ($m_a = 1$) until a total of ten nodes is attained.

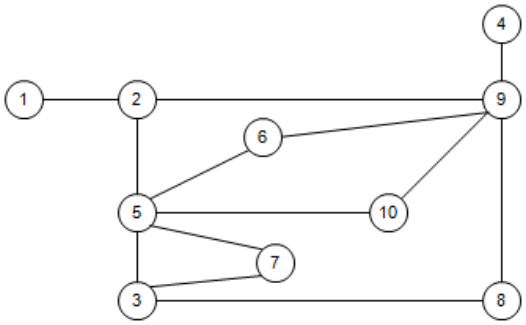


Figure 34. Example of an ER network

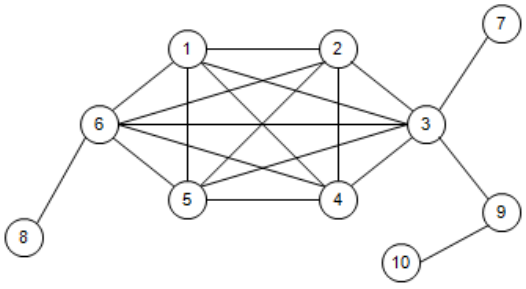


Figure 35. Example of a BA network

Watts-Strogatz (WS) networks are constructed by beginning with a lattice structure of nodes and 'rewiring' nodes randomly [64]. Figure 36 displays a WS network consisting of ten nodes in a lattice having parameters $k = 4$ and $p = 0.1$, indicating that each node is initially connected to four others (two in each direction along the lattice) and ten percent of the initial arcs are rewired.

Alternatively, Morris *et al.* developed a network generator, the Prescribed Node Degree, Connected Graph (PNDCG), that takes as an input the number of nodes, a degree distribution, and a parameter indicating the amount of clustering desired. The algorithm generates a connected network with the desired degree distribution. Extensions of the algorithm allow altering the amount of clustering and degree correlation in the network [51]. The network in Figure 37 was constructed for ten nodes having an equally likely probability (20%) of having degree one, two, three, four, or five, and a desired clustering of 100 percent ($\beta = 1$) utilizing the generator. The ER networks are generated using the PNDCG generator with a input of the binomial distribution for the parameter p as the degree distribution and clustering on or off as indicated by $\beta = 1$ or $\beta = 0$, respectively. This ensures that the ER networks are connected, rather than testing for connectivity before accepting a randomly generated network otherwise.

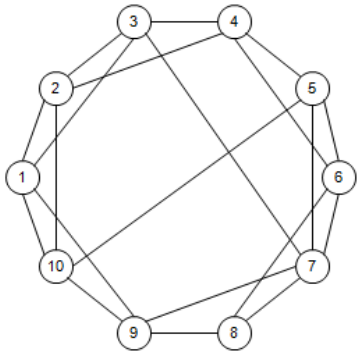


Figure 36. Example of a WS network

The test bank includes networks with about 100 nodes or roughly 1,000 nodes. Each level of network size has a low and high density (number of arcs). The density γ of a network is given by

$$\gamma = \frac{2m}{n(n-1)}.$$

Networks with density levels less than 0.15 (low-density) and more than 0.30 (high-density) were utilized within the test bank. The grid and star-mesh network structures populate the low-density networks, while the random networks were used across all density levels. The appropriate parameters will be used to ensure correct sizes and densities for the described experiments when generating the networks. The test bank was used to evaluate each of the methods. The appropriate summary statistics for solution times was reported. Any additional measures appropriate for the given test was reported as well.

Table 61 enumerates the parameter settings for the randomly generated networks. For each network size, the appropriate parameters are given for low- and high-density networks. For each type of network in the test plan, an adjacency matrix was generated. For testing maximum flow and shortest path models, source and terminus nodes were selected. The source node for all network instances is node 1. The terminus is node n, where n is the number of nodes in the instance, for grid, star-mesh, ER, and PNDCG networks. In WS

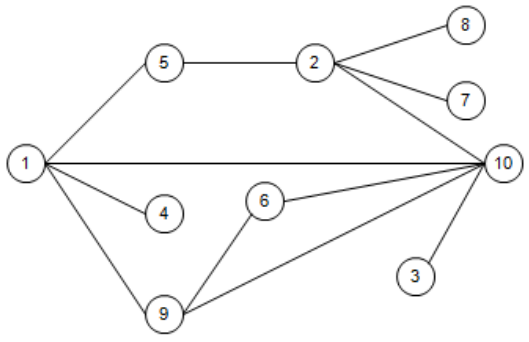


Figure 37. Example of a PNDCG network

Table 61. Random Network Parameters

		$n \approx 100$		$n \approx 1,000$	
		$\gamma < 0.15$	$\gamma > 0.30$	$\gamma < 0.15$	$\gamma > 0.30$
Grid	Square	$h = 10$ $v = 10$	N/A	$h = 32$ $v = 32$	N/A
	Tall Rectangle	$h = 5$ $v = 20$	N/A	$h = 10$ $v = 100$	N/A
	Wide Rectangle	$h = 20$ $v = 5$	N/A	$h = 100$ $v = 10$	N/A
Star-Mesh	Even	$r = 10$ $c = 10$	N/A	$r = 32$ $c = 32$	N/A
	More Rings	$r = 5$ $c = 20$	N/A	$r = 10$ $c = 100$	N/A
	More Rays	$r = 20$ $c = 5$	N/A	$r = 100$ $c = 10$	N/A
ER	$\beta = 0$	$p = 0.1$	$p = 0.5$	$p = 0.1$	$p = 0.5$
	$\beta = 1$	$p = 0.1$	$p = 0.5$	$p = 0.1$	$p = 0.5$
BA	$m_a = 2$	$n_0 = 25$	$n_0 = 55$	$n_0 = 250$	$n_0 = 550$
	$m_a = \frac{n_0}{5}$	$n_0 = 25$	$n_0 = 50$	$n_0 = 250$	$n_0 = 500$
WS	$p = 0.1$	$k = 4$	$k = 30$	$k = 40$	$k = 300$
	$p = 0.25$	$k = 4$	$k = 30$	$k = 40$	$k = 300$
PNDCG	No clustering	dist $= U_{3,5}$	dist $= U_{20,5}$	dist $= U_{3,5}$	dist $= U_{200,5}$
	$\beta = 0$	$\alpha = 2.35$	N/A	$\alpha = 2.35$	N/A
	Clustering	dist $= U_{3,5}$	dist $= U_{20,5}$	dist $= U_{3,5}$	dist $= U_{200,5}$
	$\beta = 1$	$\alpha = 2.35$	N/A	$\alpha = 2.35$	N/A

networks, node $\frac{n}{2}$ was selected as the terminus, since node n is likely connected to node 1 due to the initial lattice structure. For BA networks, the terminus is node n_0. This ensures that more than m_a paths to the terminus are available and therefore larger flow.

For each entry in the table, a random network was generated. The adjacency matrix was stored. In addition, distance, capacity, interdiction cost, and restoration time were generated

Table 62. Data Generated for Random Networks

Type	Data	Name	Size	Values
Arc	Adjacency	A	$n \times n$	-
	Distance	D	$n \times n$	Uniform(1,10)
	Capacity	C	$n \times n$	Uniform(1,10)
	Interdiction Cost	B	$n \times n$	Uniform(1,10)
	Restoration Time	R	$n \times n$	Uniform(1,10)
Node	Capacity	C_N	$n \times 1$	Uniform(1,10)
	Interdiction Cost	B_N	$n \times 1$	Uniform(1,10)
	Restoration Time	R_N	$n \times 1$	Uniform(1,10)
General	Source Node	S	1×1	1
	Terminus Node	T	1×1	n or $\frac{n}{2}$
	Connected Set	J	$\left(\frac{n}{20}, \frac{n}{10}\right) \times 1$	-
	Mission Duration	L	1×1	10

for each arc. Likewise, capacity, interdiction cost, and restoration time information was generated for each node. The values of these parameters were selected using a discrete uniform distribution with a range between 1 and 10.

Finally, for each network, a set J of connected nodes was selected (to be the divert set or another subset of nodes). An arbitrary node was selected that is not connected to the source or terminus. Subsequently, a node that is connected to the set J (and not connected to the source or terminus) is added to the set. This was repeated until there were between $\frac{n}{20}$ and $\frac{n}{10}$ nodes in the set. For BA networks, the restriction of nodes connected to the source or terminus is unattainable since such a large portion of the network is connected to the source in its construction, so nodes connected to the source or terminus were allowed in the set. In some instances of low-density networks, the size of the set is allowed to contain fewer nodes because the there were no other nodes connected to the set that could be selected.

Thus, for each network case in the test plan, a number of random networks was generated along with the parameters listed in Table 62. The randomly generated networks and associated data was stored so comparisons could be made between the networks. The mission duration L which is used for restoration time models is 10 for all networks. This

value allowed certain interdiction tasks to be effective for the duration of the time horizon considered since the largest possible restoration time was 10.

Modeling Network Interdiction Tasks

FOIL

Maj Benjamin S. Kallemyn
Committee: Dr. Richard F. Deckro (Advisor)
LTC Brian J. Lunday, PhD
Lt Col Matthew J. Robbins, PhD
Maj Dustin G. Mixon, PhD

PROBLEM STATEMENT

Given a network, identify the arcs and/or nodes to target that are most effective in destroying, diverting, disrupting, or delaying its capabilities.

DEFINITIONS

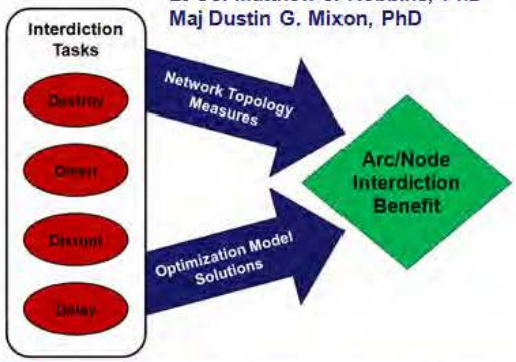
DESTROY – actions that render enemy targets ineffective or useless.
DIVERT – actions that divert enemy assets from areas they are required.
DISRUPT – actions that interrupt or impede enemy capabilities.
DELAY – actions that delay the time of arrival of enemy capabilities.
•Joint Publication 3-03, *Doctrine for Joint Interdiction Operations*.

BACKGROUND

Mission planners seek to target nodes and/or arcs in networks that have the greatest benefit for an operational plan.

An interdiction task an event that targets the nodes and/or arcs of a network resulting in its capabilities being destroyed, diverted, disrupted, or delayed.

Lessons learned from studying network interdiction model outcomes help to inform attack strategies. Conversely, the same models can be used to help identify critical nodes and/or arcs in networks that must be defended and/or hardened.



Task (arc capacities, lengths = 1)	Original Network	Attack Strategy	Residual Network
Destroy (targeted arc)	max flow = 2; min path = 3	interdiction cost = 2	max flow = 0; min path = 0
Divert (no flow available)	max flow = 2; min path = 3	interdiction cost = 2	max flow = 1; min path = 4
Disrupt (disrupted arc)	max flow = 2; min path = 3	interdiction cost = 2	max flow = 1; min path = 4
Delay (restoration)	total max flow for duration = 4	interdiction cost = 2	total max flow for duration = 2

RESEARCH OBJECTIVE

Develop a suite of network models and measures that will assist decision makers in identifying critical nodes and/or arcs to support deliberate and rapid planning and analysis.

The interdiction benefit of a node or arc is a measure of the impact an interdiction task against it has on the residual network.

The nodes and/or arcs with the largest benefit are those that should be targeted when developing an attack strategy against a network or defended (and/or hardened) when developing a defense strategy.

CONTRIBUTIONS

A suite of tools which includes:

- An algorithm to compute and update network measures when a node is targeted for destruction.
- A framework with which to model the following network interdictions tasks:
  - destroying up to *k* arcs,
  - diverting network resources from traversing through any of a predefined set of nodes,
  - disrupting network capabilities based on partial damage, and
  - delaying the restoration of network resources.

Department of Operational Sciences

Bibliography

[1] *Doctrine for Joint Interdiction Operations*. Joint Publication 3-03, Department of Defense, October 14, 2011.

[2] "Israel Strikes Symbols of Hamas' Control in Gaza, Shuts down Power Plant", *FoxNews.com*, Published: July 29, 2014. (Retrieved July 29, 2014).

[3] Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice hall, 1993.

[4] Barabási, Albert-László and Réka Albert. "Emergence of scaling in random networks", *science*, 286(5439):509–512, 1999.

[5] Bard, Jonathan F. *Practical bilevel optimization: algorithms and applications*. Springer, 1998.

[6] Bazaraa, Mokhtar S., John J. Jarvis, and Hanif D. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2005.

[7] Bellmore, M., G. Bennington, and S. Luhore. "A network isolation algorithm", *Naval Research Logistics Quarterly*, 17(4):461–469, 1970.

[8] Black, Paul E. ""sparse graphs"". in *Dictionary of Algorithms and Data Structures* [online], Vreda Pieterse and Paul E. Black eds., 14 August 2008. Available from: http://www.nist.gov/dads/HTML/rootedtree.html (accessed 19 March 2015).

[9] Bracken, Jerome and James T. McGill. "Mathematical programs with optimization problems in the constraints", *Operations Research*, 21(1):37–44, 1973.

[10] Brandes, Ulrik. "A faster algorithm for betweenness centrality", *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[11] Brandes, Ulrik. "On variants of shortest-path betweenness centrality and their generic computation", *Social Networks*, 30(2):136–145, 2008.

[12] Chartrand, Gary, Linda Lesniak, and Ping Zhang. *Graphs & digraphs*. CRC Press, 2010.

[13] Chartrand, Gary and Songlin Tian. "Distance in Digraphs", *Computers & Mathematics with Applications*, 34(11):15 – 23, 1997.

[14] Cintron-Arias, Ariel, Norman Curet, Lisa Denogean, Robert Ellis, Corey Gonzalez, Shobha Oruganti, and Patrick Quillen. "A Network Diversion Vulnerability Problem", 2000.

[15] Collado, Ricardo A. and David Papp. *Network interdiction–models, applications, unexplored directions*. Technical report, Rutcor Research Report, 2012.

[16] Corley, H.W. and David Y. Sha. "Most vital links and nodes in weighted networks", *Operations Research Letters*, 1(4):157–160, 1982.

[17] Corley Jr, H.W. and Han Chang. "Finding the n most vital nodes in a flow network", *Management Science*, 21(3):362–364, 1974.

[18] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

[19] Cullenbine, Christopher A., R. Kevin Wood, and Alexandra M. Newman. "Theoretical and computational advances for network diversion", *Networks*, 2013. Forthcoming.

[20] Curet, Norman D. "The network diversion problem", *Military Operations Research*, 6(2):35–44, 2001.

[21] Curet, Norman D., Jason DeVinney, and Matthew E. Gaston. "An efficient network flow code for finding all minimum cost *s-t* cutsets", *Computers & Operations Research*, 29(3):205–219, 2002.

[22] Dempe, Stephan. *Foundations of bilevel programming*. Springer, 2002.

[23] Ding, Chris HQ, Xiaofeng He, and Hongyuan Zha. "A spectral method to separate disconnected and nearly-disconnected web graph components". *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 275–280. ACM, 2001.

[24] Ehrgott, Matthias. *Multicriteria optimization*, volume 2. Springer, 2005.

[25] Erdős, Paul and Alfréd Rényi. "On random graphs", *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[26] Erken, Ozgur. *A branch-and-bound algorithm for the network diversion problem*. Ph.D. thesis, Monterey, California. Naval Postgraduate School, 2002.

[27] Fiedler, Miroslav. "Algebraic connectivity of graphs", *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[28] Floyd, Robert W. "Algorithm 97: Shortest Path", *Commun. ACM*, 5(6):345, June 1962. ISSN 0001-0782.

[29] Ford, L.R. and D.R. Fulkerson. *Flows in networks*, volume 3. Princeton University Press, 1962.

[30] Freeman, Linton C. "Centrality in social networks conceptual clarification", *Social networks*, 1(3):215–239, 1979.

[31] Fulkerson, D Ray and Gary C Harding. "Maximizing the minimum source-sink path subject to a budget constraint", *Mathematical Programming*, 13(1):116–118, 1977.

[32] Gergana. "Octave Networks Toolbox First Release", July 2014.

[33] Ghare, PM, Douglas C Montgomery, and WC Turner. "Optimal interdiction policy for a flow network", *Naval Research Logistics Quarterly*, 18(1):37–45, 1971.

[34] Gimbert, Joan and Nacho López. "On the construction of radially Moore digraphs.", *Ars Comb.*, 110:455–467, 2013.

[35] Golden, Bruce. "A problem in network interdiction", *Naval Research Logistics Quarterly*, 25(4):711–713, 1978.

[36] Gorman, Sean P. and Edward J. Malecki. "The networks of the Internet: an analysis of provider networks in the USA", *Telecommunications Policy*, 24(2):113–134, 2000.

[37] Gross, Jonathan L. and Jay Yellen. *Handbook of graph theory*. CRC press, 2004.

[38] Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart. "Exploring network structure, dynamics, and function using NetworkX". *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 11–15. Pasadena, CA USA, August 2008.

[39] for Health Statistics (US), National Center. "Health, United States, 2009: With special feature on medical technology", Jan 2010. Available from: http://www.ncbi.nlm.nih.gov/books/NBK44737/.

[40] Herbranson, Travis J., Richard F. Deckro, James W. Chrissis, and Jonathan Todd Hamill. "Considering the isolation set problem", *European Journal of Operational Research*, 227(2):268 – 274, 2013.

[41] Herodotus. *The Histories*. Trans. A. D. Godley (Cambridge: Harvard University Press, 1920), Perseus Digital Library. http://www.perseus.tufts.edu (Retrieved July 24, 2014).

[42] Israeli, Eitan and R. Kevin Wood. "Shortest-path network interdiction", *Networks*, 40(2):97–111, 2002.

[43] Jackson, Richard H.F., Paul T. Boggs, Stephen G. Nash, and Susan Powell. "Guidelines for reporting results of computational experiments. Report of the ad hoc committee", *Mathematical programming*, 49(1):413–425, 1990.

[44] Kallemyn, Benjamin S., Richard F. Deckro, and Kevin T. Kennedy. "On Considering multiple optimal solutions in network interdiction", 1–15, 2015. Working paper.

[45] Kennedy, Kevin T., Richard F. Deckro, James T. Moore, and Kenneth M. Hopkinson. "Nodal interdiction", *Mathematical and Computer Modelling*, 54(11):3116–3125, 2011.

[46] Knor, Martin. "A note on radially Moore digraphs", *Computers, IEEE Transactions on*, 45(3):381–382, 1996.

[47] Malik, Kavindra, A.K. Mittal, and Santosh K. Gupta. "The k most vital arcs in the shortest path problem", *Operations Research Letters*, 8(4):223–227, 1989.

[48] Mohar, Bojan and Y Alavi. "The Laplacian spectrum of graphs", *Graph theory, combinatorics, and applications*, 2:871–898, 1991.

[49] Mohar, Bojan and Tomaž Pisanski. "How to compute the Wiener index of a graph", *Journal of Mathematical Chemistry*, 2(3):267–277, 1988.

[50] Montgomery, Maggie A and Menachem Elimelech. "Water and sanitation in developing countries: including health in the equation", *Environmental Science & Technology*, 41(1):17–24, 2007.

[51] Morris, James F, Jerome W O'Neal, and Richard F Deckro. "A random graph generation algorithm for the analysis of social networks", *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 1548512912450370, 2013.

[52] Nardelli, Enrico, Guido Proietti, and Peter Widmayer. "Finding the most vital node of a shortest path", *Theoretical computer science*, 296(1):167–177, 2003.

[53] Newman, Mark. *Networks: an introduction*. Oxford University Press, 2010.

[54] Newman, Mark E.J., Steven H. Strogatz, and Duncan J. Watts. "Random graphs with arbitrary degree distributions and their applications", *Physical review E*, 64(2):026118, 2001.

[55] Plesník, Ján. "On the sum of all distances in a graph or digraph", *Journal of Graph Theory*, 8(1):1–21, 1984.

[56] Rocco S., Claudio M., José Emmanuel Ramirez-Marquez, and Daniel E. Salazar A. "Bi and tri-objective optimization in the deterministic network interdiction problem", *Reliability Engineering & System Safety*, 95(8):887–896, 2010.

[57] Royset, Johannes O. and R. Kevin Wood. "Solving the Bi-Objective Maximum-Flow Network-Interdiction Problem", *INFORMS Journal on Computing*, 19(2):175–184, 2007.

[58] Sherali, Hanif D. "Equivalent weights for lexicographic multi-objective programs: characterizations and computations", *European Journal of Operational Research*, 11(4):367–379, 1982.

[59] Sherali, Hanif D. and Allen L. Soyster. "Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples", *Journal of Optimization Theory and Applications*, 39(2):173–186, 1983.

[60] Department of Transportation, Federal Highway Administration. "Final Map of The Eisenhower Interstate System", *www.fhwa.dot.gov/interstate/finalmap.cfm*. Retrieved January 30, 2014.

[61] Valente, Thomas W. and Robert K. Foreman. "Integration and radiality: Measuring the extent of an individual's connectedness and reachability in a network", *Social Networks*, 20(1):89 – 105, 1998.

[62] Warshall, Stephen. "A Theorem on Boolean Matrices", *J. ACM*, 9(1):11–12, January 1962. ISSN 0004-5411.

[63] Wasserman, Stanley and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[64] Watts, Duncan J. and Steven H. Strogatz. "Collective dynamics of small-worldnetworks", *nature*, 393(6684):440–442, 1998.

[65] West, Douglas Brent et al. *Introduction to graph theory*, volume 2. Prentice Hall, Upper Saddle River, 2001.

[66] Wiener, Harry. "Structural determination of paraffin boiling points", *Journal of the American Chemical Society*, 69(1):17–20, 1947.

[67] Wollmer, Richard. "Removing arcs from a network", *Operations Research*, 12(6):934–940, 1964.

[68] Wood, R. Kevin. "Deterministic network interdiction", *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[69] Wood, R. Kevin. *Bilevel Network Interdiction Models: Formulations and Solutions*. John Wiley & Sons, Inc., 2010.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED *(From — To)*
17-09-2015	Dissertation	AUG 2012 – SEP 2015

4. TITLE AND SUBTITLE

Modeling Network Interdiction Tasks

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kallemyn, Benjamin S., Major, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENS-DS-15-S-032

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved For Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Mission planners seek to target nodes and/or arcs in networks that have the greatest benefit for an operational plan. In joint interdiction doctrine, a top priority is to assess and target the enemy's vulnerabilities resulting in a significant effect on its forces.

An interdiction task is an event that targets the nodes and/or arcs of a network resulting in its capabilities being destroyed, diverted, disrupted, or delayed. Lessons learned from studying network interdiction model outcomes help to inform attack and/or defense strategies.

A suite of network interdiction models and measures is developed to assist decision makers in identifying critical nodes and/or arcs to support deliberate and rapid planning and analysis. The interdiction benefit of a node or arc is a measure of the impact an interdiction task against it has on the residual network.

The research objective is achieved with a two-fold approach. The measures approach begins with a network and uses node and/or arc measures to assess the benefit of each for interdiction. Concurrently, the models approach employs optimization models to explicitly determine the nodes and/or arcs that are most important to the planned interdiction task.

15. SUBJECT TERMS
Network Interdiction, Destroy, Divert, Disrupt, Delay, Optimization Models

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	259	Richard F. Deckro, AFIT/ENS
					19b. TELEPHONE NUMBER (Include Area Code) (937) 255-6565, ext 4325 (richard.deckro@afit.edu)

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18